Implementação e Análise de BRDFs Difusas e Especulares no contexto de *Path Tracing* Interativo

Victor Sales Dantas

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2019

Victor Sales Dantas

Implementação e Análise de BRDFs Difusas e Especulares no contexto de *Path Tracing* Interativo

Monografia apresentada ao curso Engenharia da Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em titulo

Orientador: Christian Azambuja Pagot

Coorientador: Andrei de Araujo Formiga

Junho de 2019

Catalogação na publicação
Seção de Catalogação e Classificação

D192i Dantas, Victor Sales.
Implementação e Análise de BRDFs Difusas e Especulares no contexto de Path Tracing Interativo / Victor Sales Dantas. - João Pessoa, 2019.
86 f. : il.

Orientação: Christian Azambuja Pagot.
Coorientação: Andrei de Araujo Formiga.
Monografia (Graduação) - UFPB/De informática.

1. BRDF. 2. Path Tracing. 3. PBR. 4. Fotorrealismo. 5. Custo computacional. 6. Performance. 7. Tempo real. 8. GPU. I. Pagot, Christian Azambuja. II. Formiga, Andrei de Araujo. III. Título.

UFPB/CI

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Engenharia da Computaçãointitulado ***Implementação e Análise de BRDFs Difusas e Especulares no contexto de Path Tracing Interativo*** de autoria de Victor Sales Dantas, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Liliane dos Santos Machado
Centro de Informática da Universidade Federal da Paraíba (CI/UFPB)

Prof. Dr. Álisson Vasconcelos de Brito
Centro de Informática da Universidade Federal da Paraíba (CI/UFPB)

Coordenador(a) do Centro de Informática
Fernando Menezes Matos
CI/UFPB

João Pessoa, 4 de junho de 2019

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

DEDICATÓRIA

“ - Agora o senhor dorme primeiro, Sr. Frodo - disse ele. – Está ficando escuro de novo. Calculo que este dia esteja quase terminado.

Frodo suspirou e adormeceu quase antes de Sam terminar suas palavras. Sam lutava contra o próprio cansaço, e segurou a mão de Frodo, e assim, sentado, até que a noite profunda caiu. Então, por fim, para manter-se acordado, saiu do esconderijo e ficou observando. A região parecia cheia de estalos, rangidos e ruídos dissimulados, mas não havia som de vozes ou passos. Bem acima dos Ephel Dúath, no oeste, o céu noturno estava Pálido e baço. Lá, espiando por entre os restos de nuvens sobre uma rocha pontiaguda nas montanhas, Sam viu uma estrela branca reluzir por uns momentos. Sua beleza arrebatou-lhe o coração, quando desviou os olhos da terra desolada, e ele sentiu a esperança retornar. Pois como um raio, cristalino e frio, invadiu-o o pensamento de que afinal de contas a Sombra era apenas uma coisa pequena e passageira: havia luz e uma beleza nobre que eram eternas e estavam além do alcance dela. A canção que cantara na torre fora mais um desafio que uma esperança, pois naquela hora pensara em si mesmo. Agora, por um momento, sua própria sorte, e até a de seu mestre, deixaram de preocupá- lo. Sam voltou às sarças e se deitou ao lado de Frodo, e, deixando de lado todo o medo, mergulhou num sono profundo e despreocupado”[2]

“No princípio era o Verbo, e o Verbo estava com Deus, e o Verbo era Deus. Ele estava no princípio com Deus. Todas as coisas foram feitas por ele, e sem ele nada do que foi feito se fez. Nele estava a vida, e a vida era a luz dos homens. E a luz resplandece nas trevas, e as trevas não a compreenderam. Houve um homem enviado de Deus, cujo nome era João. Este veio para testemunho, para que testificasse da luz, para que todos cressem por ele. Não era ele a luz, mas para que testificasse da luz. Ali estava a luz verdadeira, que ilumina a todo o homem que vem ao mundo. Estava no mundo, e o mundo foi feito por ele, e o mundo não o conheceu. Veio para o que era seu, e os seus não o receberam. Mas, a todos quantos o receberam, deu-lhes o poder de serem feitos filhos de Deus, aos que crêem no seu nome; Os quais não nasceram do sangue, nem da vontade da carne, nem da vontade do homem, mas de Deus. E o Verbo se fez carne, e habitou entre nós, e vimos a sua glória, como a glória do unigênito do Pai, cheio de graça e de verdade”(João 1,1-14).

Dedico este documento à luz eterna.

AGRADECIMENTOS

Agradeço e dou graças primeiramente a Deus, com as mesmas palavras do salmista: "Vós convertestes o meu pranto em prazer, tirastes minhas vestes de penitência e me cingistes de alegria. Assim, minha alma vos louvará sem calar jamais"(Salmo 29). Sempre serei grato aos meus pais (Maria de Fátima e João Batista), minhas irmãs (Samara Sales e Iasmim Sales), à minha namorada (Letícia Medeiros) e à minha família, por todo esforço, todo apoio nos piores momentos, sempre renovando minhas forças e me motivando a continuar e não desistir dos meus objetivos. Pelo mesmo motivo, agradeço também aos meus amigos, para os quais partilhei minhas conquistas, minhas frustrações, onde encontrei sempre conforto, alegria e consolo nos mais diversos momentos da minha vida (Andrea Brito, Chaenne Pessoa, Sávio Fonseca, Richelieu Costa, Clara Rezende, Bela Rezende, Lucas Abreu, Clara Alves, Deborah Oliveira, João Pedro Antunes, Guilherme Antunes, Davi Antunes, Israel Alves, Brenda Lucena, Rafael Machado, José Alves, Gabriel Gonçalves, Lucas Lacerda, Lucas Vieira, Roger Peixoto, Victor Moraes, Emmanuel Franco, João Alberto e todos aqueles que me acompanharam estes anos de graduação e que, por descuido, esqueci de citar o nome). Agradeço, também, aos meus orientadores (Christian Pagot e Andrei Formiga) por todo auxílio, paciência e sabedoria durante o processo de desenvolvimento deste trabalho e agradeço, de modo especial, à Christopher Régis, que, junto aos meus orientadores, foi um grande colaborador deste trabalho, me ajudando a compreender muitos dos conceitos necessários para implementação do *Path Tracing* e das BRDFs em GPU.

RESUMO

O algoritmo de *Path Tracing* [39] é um método de síntese de imagem fotorrealista considerado estado da arte em métodos de Renderização Baseado em Física (PBR) [11], também conhecido na computação gráfica como um método que demanda alto custo computacional. Em decorrência disto, estruturas de aceleração [13][42], técnicas de refinamento de amostras [32][34] e a programação paralela [40], tornam-se instrumentos indispensáveis para otimizá-lo [11]. Dentro do *Path Tracing*, existem maneiras diferentes de calcular o brilho dos materiais existentes na natureza. O principal objetivo deste estudo é fazer uma análise dos modelos das Funções de Distribuição de Reflectância Bidirecional (BRDF) [18], equações responsáveis por modelar estes materiais, de modo a comparar a performance e a semelhança das imagens geradas pelos mesmos. Os modelos analisados neste estudo são os modelos difusos, de Lambert [20] e de Oren-Nayar [26], e os modelos especulares, de Phong [29] e de Cook-Torrance [31]. Esta comparação tem a finalidade de elencar qual modelo difuso e qual modelo especular é adequado para implementação, levando em consideração as situações de conflito de escolha entre performance e fotorrealismo, questão relevante e recorrente dentro do contexto do algorítmo em tempo real na GPU. Este estudo, por apresentar de forma mais detalhada os elementos do algoritmo de *Path Tracing*, uma visão geral de algumas técnicas de aceleração do mesmo e a descrição e a avaliação de performance de alguns modelos de BRDF, é uma boa fonte de consulta para aqueles que pretendem implementar os modelos apresentados, devido ao número de referências e às discussões citadas no desenvolver deste documento.

Palavras-chave: BRDF, *Path Tracing*, PBR, Fotorrealismo, Custo computacional, Performance, Tempo real, GPU.

ABSTRACT

The Path Tracing [39] algorithm is a photorealistic image synthesis method considered state of the art in physics-based rendering (PBR) methods, also known in computer graphics as a method that demands high computational cost. As a result of this, acceleration structures [13][42], sample refinement techniques [32][34] and parallel programming [40], become indispensable tools to optimize it [11]. Inside the Path Tracing, there are different ways of calculating the brightness of the existing materials in nature. The main objective of this study is to analyze the models of the Bidirectional Reflectance Distribution Functions (BRDF) [18], equations responsible for modeling these materials, in order to compare the performance and similarity of the images generated by them. The models analyzed in this study are the diffuse models, Lambert [20] and Oren-Nayar [26], and the specular models, Phong [29] and Cook-Torrance [31]. This comparison has the purpose of listing which diffuse model and which specular model is suitable for implementation, taking into account the situations of conflict of choice between performance and photorealism, relevant and recurrent question within the context of the real-time algorithm in GPU. This study, by presenting in more detail the elements of the Path Tracing algorithm, an overview of some acceleration techniques and the description and the evaluation of performance of some BRDF models, is a good source of consultation for those who intend to implement the presented models, due to the number of references and discussions cited in the development of this document.

Keywords: BRDF, *Path Tracing*, PBR, Photorealism, Computational cost, Performance, Real time, GPU.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

SIGLA – NOME COMPLETO

SI – Sistema Internacional de Unidades

BRDF - Função de Distribuição de Reflectância Bidirecional

PBR - Renderização Baseada em Física

GPU - Unidade de Processamento Gráfico

CPU - Unidade Central de Processamento

PDF - Função de Distribuição de Probabilidade

FDN - Função de Distribuição de Normais

GAF - Termo de Atenuação Geométrico

Sumário

1 INTRODUÇÃO

A indústria de um modo geral - seja ela automobilística, de cinema ou de jogos - busca sempre aprimorar o seu produto para estar à frente da concorrência no mercado. No caso da indústria de jogos, um dos critérios de avaliação de qualidade de um produto é a quantidade de quadros por segundo e o realismo da imagem gerada a cada quadro (fotorrealismo e performance) [41]. Existem, na computação gráfica, diferentes métodos utilizados no processo de síntese de imagem, dentre eles destaca-se o algoritmo de *Path Tracing* [39], que simula o transporte de energia luminosa, gerando resultados realistas a ponto de serem semelhantes a fotografias. Embora seja o estado da arte na síntese de imagens fotorrealistas, este algoritmo é muito caro computacionalmente, o que é um empecilho para sua utilização em aplicações em tempo real, como jogos. Ao longo dos anos, a pesquisa científica envolvendo esta área buscou otimizar este processamento com programação paralela, instruções vetoriais [48], através da programação da GPU [40] pelo uso do máximo de unidades de processamento paralelo, possibilitado pela arquitetura *multicore* da mesma, e estruturas de dados de aceleração [42], para que pudesse tornar o *Path Tracing* uma boa alternativa para as diversas aplicabilidades da computação gráfica e da síntese de imagens em tempo real. Para sintetizar imagens semelhantes a fotografias utilizando o algorítmo e simulando diversos materiais existentes na natureza, é necessário descrever algebricamente uma grande variedade de tipos de materiais, como eles reagem à incidência de energia luminosa de modo a representar materiais mais semelhante à vida real possível. A grande dificuldade é representar matematicamente uma infinidade de variações de tipos de materiais como eles existem, sem que demande alto custo computacional e queda da performance da aplicação em tempo real.

Como demanda tempo estudar e implementar os diversos modelos de materiais, mesmo os mais difundidos da literatura da computação gráfica, e não se sabe qual a melhor maneira de representar um tipo de material específico, visto que existem modelos diferentes, formas diferentes de descrever analiticamente estes modelos no contexto do Path Tracing, considerando o realismo dos resultados gerados pelos modelos e o custo computacional do mesmo). Já que a escolha do modelo a ser implementado não é uma tarefa simples e custa muito tempo, o objetivo do presente estudo é fazer uma comparação, no contexto do *Path Tracing* em tempo real (na GPU), de dois modelos de BRDF difusas (de Lambert [20] e de Oren-Nayar [26]) e dois modelos de BRDF especulares (de Phong [29] e de Cook-Torrance [31]), amplamente utilizados e discutidos na literatura da computação gráfica [17][18][25], a fim de discutir, avaliando a performance e o realismo dos resultados obtidos com estes modelos, para que seja possível sugerir qual modelo difuso e qual modelo especular é mais adequado para implementação, dependendo da situação. Para isto, se fez necessário introduzir, neste trabalho, não só as BRDFs, mas também alguns conceitos para compreensão do *Path Tracing*: como ele funciona, como simula o transporte

de energia, quais são as características do algorítmo que o tornam computacionalmente custoso e as otimizações mais conhecidas para estas características. Desta forma, este trabalho contempla áreas de conhecimento da física, como a radiosidade [23][4][14], como também da estatísitca e da probabilidade [3], além de explicar a importância da GPU no funcionamento dos algorítmos de síntese de imagens interativos [3].

2 FUNDAMENTAÇÃO TEÓRICA

Esta sessão visa introduzir uma série de conceitos necessários para a compreensão do trabalho. São eles o *Path Tracing*, as técnicas de aceleração que existem na literatura para o mesmo, e alguns conceitos introdutórios sobre a GPU e sua relação com a otimização do algorítmo estudado.

Como foi dito anteriormente, o *Path Tracing* é um método de renderização baseado em física, isto é, um método de síntese de imagem que busca simular, com maior precisão possível, a forma que a energia luminosa se transporta no meio em que está inserida, modelando equações que descrevam estes fenômenos físicos.

2.1 Princípios radiométricos

Antes de introduzir a equação de rendering, de demonstrar algumas simplificações e explicar como a equação de rendering pode ser resolvida por um método numérico - tema das próximas sessões - é necessário que haja um breve estudo de algumas quantidades radiométricas que são fundamentais para o entendimento e implementação da equação de rendering. Radiometria signifca, como o próprio nome diz, é a área da ciência que estuda as quantidades radiométricas, o que inclui o espectro da luz visível pelo olho humano. Para estudar radiometria nesta sessão, serão utilizadas as unidades de medida do sistema internacional de unidades(SI).

Comprimento de onda

A luz é estudada em radiometria como uma onda eletromagnética, como em [23]. Ondas eletromagnéticas possuem uma propriedade chamada comprimento de onda (λ) que é a distância entre duas cristas ou dois vales. O comprimento de uma onda eletromagnética pode assumir um valor qualquer dentro de um intervalo contínuo e muito grande, o espectro eletromagnético. Porém, dentro desse espectro, existe uma faixa visível ao olho humano, apenas um intervalo menor de todo espectro eletromagnético, entre 400 nm(ultravioleta) e 780 nm(infravermelho), aproximadamente. O espectro visível da luz pode ser visto na figura a seguir.

Figura 1: Espectro visível da luz.

Potência

Potência é um fluxo de energia emitida por uma fonte por unidade de tempo, medida em $Watts$ (W) - uma abreviação da unidade de medida *joules* por segundo do SI-. Se uma lâmpada de $100W$ é ligada, a energia emitida para o ambiente será menor ou igual a 100 J/s, onde a quantidade de energia emitida para o ambiente seria menor do que 100 J/s, se fosse considerada a perda de energia em forma de calor (efeito *joule*). Para maiores informações a respeito da potência e sua relação com efeito Joule, consultar [44].

Irradiância

Esta quantidade radiométrica representa a quantidade de potência incidente por àrea, sua unidade de medida é dada por W/m^2. Para calculá-la, será utilizada uma função de densidade. A Irradiância espectral (E) é a potência $(d\Phi_\lambda)$ espectral dividida pela área (dA), sendo calculada como o exposto na seguinte equação:

$$E = \frac{d\Phi_\lambda}{dA} \tag{1}$$

Esta equação retornara o fluxo de fótons que passam por um diferencial de área, o que não é suficiente, porque será necessária uma quantidade radiométrica que expresse a contribuição de energia dos fótons que vão em direção ao ponto representado pelo diferencial de área, por enquanto, estamos levando em consideração apenas o fluxo de fótons em todas as direções, quando se busca uma quantidade radiométrica que inclua a direção de um raio de luz, para poder simular o transporte da energia em uma direção, percorrendo um caminho.

Ainda considerando a equação 1, com os diferenciais de área dA_1, se fizessemos uma medida de Irradiância E_1 para aquele ponto, teríamos um valor x em W/m^2. Numa segunda medida(E_2), para o mesmo ponto, se fosse feita uma segunda medida, aplicando um fator de escala que duplicasse o volume do sólido de revolução da primeira medida, duplicando, consequentemente, a área diferencial - e sabendo que a área diferencial é inversamente proporcional à Irradiância -, E_2 mediria um valor y em W/m^2 menor do que x($\frac{1}{4}x$). Esta proporção pode ser melhor compreendida através da figura 2.

Intensidade radiante

O Ângulo sólido ($\boldsymbol{\omega}$) é uma direção medida em esferorradianos (sr) muito utilizada para expressar outras quantidades radiométricas. No caso da figura acima, destacam-se duas áreas ($A1$ e $A2$) e duas distâncias ao mesmo ponto ($r1$ e $r2$). Imaginando que do

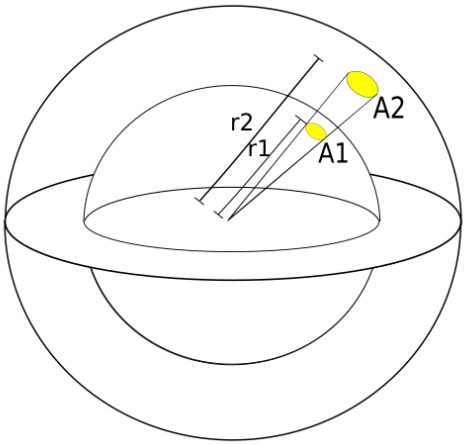


Figura 2: Ângulo sólido.

ponto de vista do centro das duas esferas o centro das duas áreas circulares destacados em amarelo sejam concêntricas e diferenciais (muito pequenas), elas representam o mesmo esferorradiano ω, chegando na equação a seguir.

$$\boldsymbol{\omega} = \frac{A_1}{r_1^2} = \frac{A_2}{r_2^2}.$$

A intensidade radiante é a quantidade de potência dissipada ($d\Phi$), ou emitida por uma fonte de energia diferencial na direção de um esferorradiano ($d\boldsymbol{\omega}$). Por exemplo: Uma fonte de luz pontual, de $100W$ de potência, distribui esta potência, de forma ideal, na mesma quantidade em todas as direções. Dado um esferorradiano que seja a metade da superfície de uma esfera (que representaria todas as possíveis direções de emissão de luz), que tem esta luz pontual de $100W$ em seu centro, a intensidade radiante neste esferorradiano deveria ser proporcional à metade da potência dissipada pela luz ($50W$). A intensidade radiante é dada pela equação:

$$I = \frac{d\Phi}{d\boldsymbol{\omega}} \tag{2}$$

Sua unidade no SI é Watts por esferorradiano (W/sr). Para esclarecer melhor os conceitos de esferorradianos aplicados aos da intensidade radiante, observe a figura a 3, que é uma representação em 2D de todas as possíveis direções que uma fonte de luz pontual pode emitir um raio de luz.

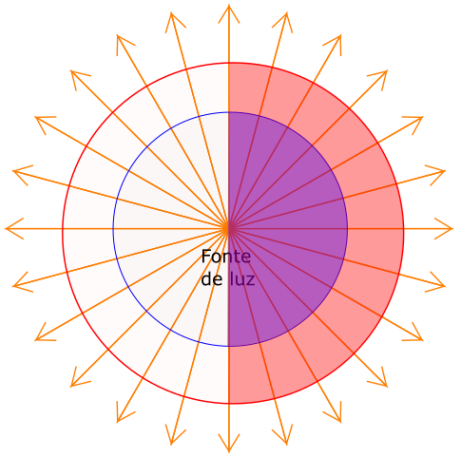


Figura 3: A fonte de Luz pontual é representada pelo círculo preenchida de laranja no centro da imagem, as circunferências conscêntricas , azul e vermelha, representam toda a superfície do sólido de revolução ao redor do mesmo ponto de duas distâncias diferentes do mesmo ponto. Os dois setores circulares semi-transparentes em azul e vermelho são os esferorradianos.

Observando a imagem, a quantidade de potência por área que transpassa a porção da área da superfície de revolução, destacados em azul e vermelho transparente, é a mesma (menor ou igual à metade de toda a potência emitida pela fonte de luz pontual).

Note que, se for alterado o ponto de partida do ângulo sólido (de todas as possíveis direções, representadas na figura 3 na forma de setas laranjas), ou o tamanho do da área da superfície delimitada pelo ângulo sólido, a intensidade radiante alterará seu valor.

Radiância

A radiância é quantidade radiométrica que expressa a proporção do fluxo total de potência num ponto (irradiância) que é redirecionada numa direção de ângulo sólido ($\boldsymbol{\omega}$), medida em Watts por unidade de área e esferorradiano (W/m^2sr), no SI. Ou seja, a radiância é o fluxo de potência que sai de um ponto, ou área diferencial, de qualquer objeto, seja por emissão, ou por reflexão, e atinge uma superfície receptora desta quantidade de energia luminosa emitida ou refletida pelo ponto, que depende do ângulo entre o vetor normal da direção do feixe de luz da fonte, com a direção do esferorradiano formado entre a fonte e o destino do fluxo potência emitida, como ilutra a figura a seguir:

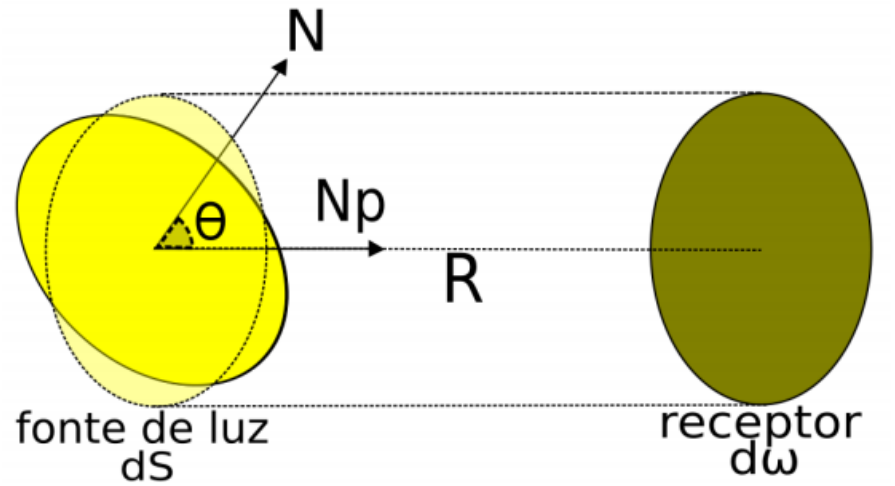


Figura 4: Radiância.

Onde dS é o diferencial de área da fonte de luz, $d\omega$ um esferorradiano diferencial, N é a direção do vetor normal, Np é a direção do esferorradiano e R a distância entre o diferencial de área projetada na direção de esferorradiano e o diferencial de esferorradiano. A equação da radiância, utilizando esta notação é expressa da seguinte maneira:

$$Li = \frac{d^2\Phi_\lambda}{dS\cos\theta_i \boldsymbol{d\omega_i}} \tag{3}$$

Como a irradiância pode ser obtida a partir de um somatório de infinitas radiâncias, para cada direção diferencial que atinge o ponto ou diferencial de área em questão (formando um hemisfério, como o sólido ω apresentado na figura 5). A equação que relaciona a Irradiância(E(x)), a partir de todas as radiâncias incidentes(Li), é dada por:

$$E(x) = \int_\Omega Li(x, \boldsymbol{\omega_i}) \cos\theta_i \ \boldsymbol{d\omega_i},$$

derivando dos dois lados da equação

$$\frac{dE(x)}{\boldsymbol{d\omega_i}} = Li(x, \boldsymbol{\omega_i}) \cos\theta_i$$

e passando o $\boldsymbol{d\omega_i}$ para o membro do lado direito da equação, temos a seguinte relação

$$dE(x) = Li(x, \boldsymbol{\omega_i}) \cos\theta_i \boldsymbol{d\omega_i} \tag{4}$$

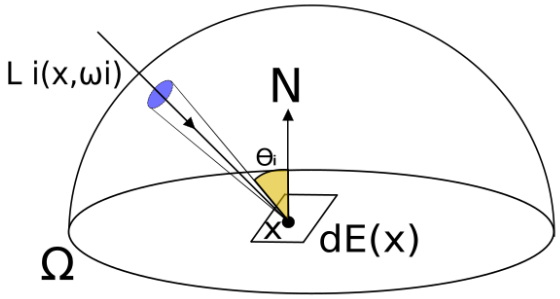


Figura 5: Irradiância diferencial(dE(x)) a partir da radiância incidente(Li).

BRDF

Para chegar na quantidade radiométrica de maior importância para nosso estudo, precisamos transformar a Irradiância($E(x, \boldsymbol{\omega_i})$) em Radiância refletida($Lr(x, \boldsymbol{\omega_o})$), ou seja, uma função fr que retorne quanto de W/m^2 (daquela irradiância), sai em direção ao meu ângulo sólido de reflexão($\boldsymbol{\omega_o}$), garantindo as propriedades físicas relacionadas ao tipo do material (como a energia se distribui em cada direção), mas também as leis físicas(por exemplo: conservação de energia). No caso dos materiais difusos, esta função é chamada BRDF (Função de distribuição de reflectância bidirecional). A relação que transforma a irradiância em radiância é dada por:

$$dE(x, \boldsymbol{\omega_i}) f(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = dLr(x, \boldsymbol{\omega_o}),$$

ou seja,

$$f(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{dLr(x, \boldsymbol{\omega_o})}{dE(x, \boldsymbol{\omega_i})} \tag{5}$$

e substituindo $dE(x, \boldsymbol{\omega_i})$ pelo segundo membro da equação 4, é possível calcular a BRDF pela proporção ilustrada na figura 6.

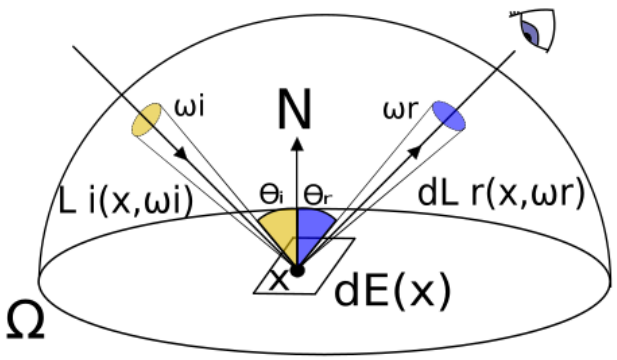


Figura 6: Diferencial de radiância refletida(dLr) a partir da radiância incidente(Li).

$$fr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{dLr(x, \boldsymbol{\omega_o})}{Li(x, \boldsymbol{\omega_i}) \cos\theta_i \boldsymbol{d\omega_i}} \tag{6}$$

2.2 A Equação de Rendering.

No ano de 1986, James T. Kajiya [39] apresentou a equação de rendering, pela primeira vez, da seguinte forma:

$$Lo = Le + Lr,$$

onde Lr é dada pela equação 7, e representa a proporção de energia refletida na direção de reflexão, dada uma direção de incidência de luz, e onde Le é a quantidade de energia emitida pela superfície, caso seja a superfície de uma fonte de luz, Le é contante e diferente de zero e Lr é a proporção de energia refletida numa direção de reflexão, dada uma direção de incidência. Para calcular a equação de reflexão, basta reorganizar os termos da equação da BRDF, expressa em 6, da seguinte forma:

$$\frac{dLr(x, \boldsymbol{\omega_o})}{\boldsymbol{d\omega_i}} = fr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) Li(x, \boldsymbol{\omega_i}) \cos\theta_i$$

Integrando os dois lados em relação a ω_i, obtemos a equação 7 (equação de reflexão).

$$Lr = \int_\Omega fr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) Li(x, \boldsymbol{\omega_i}) \cos\theta_i \ \boldsymbol{d\omega_i}, \tag{7}$$

Onde Ω é o hemisfério que comporta todos os esferorradianos diferenciais, equivalentes às direções de incidência de fluxo de potência (radiância) na nossa área diferencial dA. Em outras palavras, esta quantidade radiométrica Lr retorna quanto, em fluxo de potência, na direção da área diferencial da superfície correspondente ao ângulo sólido ω_o, considerando a contribuição energética de todas as possíveis direções de incidência de fluxo de potência por área de todo o sólido Ω (que por sua vez é todas as possíveis direções de incidência ω_i). Simplificando, quanto de energia é refletida na direção ω_o, levando em consideração todas as possíveis direções de incidência de luz (ω_i).

Para materiais que não emitem luz, a equação de rendering é igual à equação de reflexão (equação 7), mas em casos de materiais emissores de luz, precisamos acrescentar à equação da reflexão um termo que diga respeito ao fluxo de potência por diferencial de área correspondente à sua emissão, ou seja, como é ilustrado na figura 7 e expressada na equação a seguir.

Lo é a quantidade total de W/m^2, carregada pelo raio de luz, cruzando o diferencial de área do ângulo sólido, representado na figura como a direção Lr (uma dentre várias

possíveis direções de reflexões, utilizadas no algorítmo como as direções de observação), Le é a porção de Lo referente a emissão de luz e Lr é a porção de Lo equivalente a quantia de energia refletida da luz na mesma direção representada na figura (equação 7). Este valor de W/m^2 de energia refletida equivale a toda a radiância que atingem o ponto de sua origem, vinda em todas as possíveis direções em volta deste ponto (Li), formando a equação de *rendering* (equação 8).

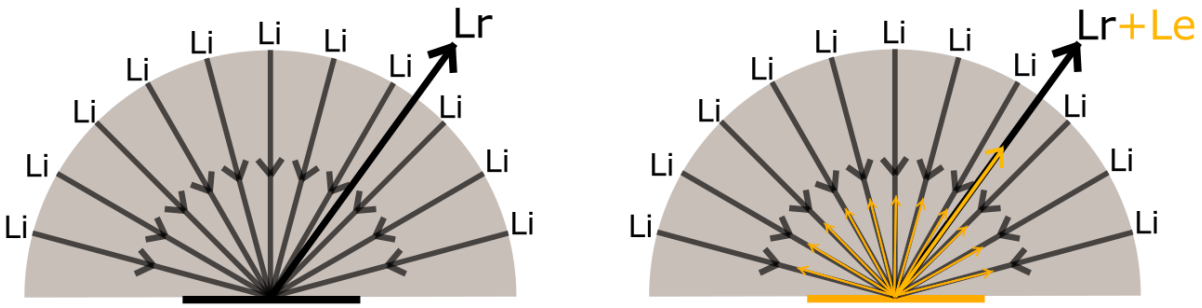


Figura 7: Material não-emissivo e emissivo.

$$Lo = Le(x, \boldsymbol{\omega_o}) + \int_{\Omega} fr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) Li(x, \boldsymbol{\omega_i}) \cos\theta_i \; \boldsymbol{d\omega_i} \tag{8}$$

A equação descrita acima descreve quanto de radiância (Lo), cuja unidade de medida é dada por $\frac{W}{m^2 sr}$, que é transportada por um raio de luz vindo do ponto x, na direção de reflexão (ω_o), levando em consideração a energia emitida por esse ponto, caso seja uma luz pontual, mas também a quantia de energia refletida de todas as possíveis direções de incidência(ω_i).

Para entender, em maior nível de detalhe, a equação de rendering e as quantidades radiométricas, consultar [4], [5] e [14].

Esta equação ainda não está na forma adequada para calcular o algorítmo de *path tracing* na GPU, pois é inviável computacionalmente resolver uma integral (soma de infinitos diferenciais discretos). Por isso, utiliza-se do artifício de métodos numéricos para resolver integrais de maneira a discretizá-las e simplificá-las. O método Monte Carlo é um dos métodos utilizados para resolver integrais e será discutido imediatamente.

2.3 Integração Monte Carlo.

Como foi explicado no capítulo "Path to Path-Traced Movies"da revista "Foundations and trends® in Computer Graphics and Vision"[1], o método Monte Carlo foi inicialmente utilizado por Enrico Fermi em meados de 1930, mas por ser um método numérico de solução de problemas matemáticos com amostragem randômica, como pode

ser visto no livro "A Primer for the Monte Carlo Method"[3], passou a ser amplamente utilizado apenas posteriormente, após a invenção de computadores eletrônicos que viabilizaram, agilisaram e automatizaram a solução destes problemas. Este método era utilizado para simular transporte de neutrons no desenvolvimento de bombas e reatores nucleares na década de 1940, já que o transporte de neutrons tem um comportamento físico da mesma natureza que o transporte de energia luminosa. Assim surgiu a aplicação do método Monte Carlo para realçar o fotorrealismo da renderização de uma cena 3D. Este método é uma alternativa probabilística para resolver integrais de funções. É necessário entender alguns conceitos de probabilidade (explicados neste capítulo) para compreender a simplificação da equação de rendering que tira proveito deste método.

Variáveis aleatórias discretas

Uma variável aleatória discreta ξ, pode assumir n valores. Para cada valor dentre os n valores que ξ pode assumir(x_i), existe uma probabilidade associada a este valor(p_i). A variável discreta, seguindo esta notação, pode ser expressa da seguinte forma:

$$\xi \sim \begin{pmatrix} x_1 & x_1 & \dots & x_n \\ p_1 & x_1 & \dots & p_n \end{pmatrix}, \tag{9}$$

onde $P\{\xi = x_i\} = p_i$.

Uma variável aleatória deve ter algumas propriedades, dentre elas, é que a soma de todas as probabilidades deve ser igual a 1.

Valor esperado de uma variável discreta

Se fizermos um experimento com uma variável aleatória e calcularmos a média aritmética de todas as amostras (para n amostras), o resultado desta média aritmética deve convergir para o valor esperado. Se a variável aleatória for um dado de n faces, cuja probabilidade de todos os eventos seja a mesma, a equação do valor esperado desta variável é dado pela equação 10. Porém, se suas probabilidades tem valores distintos entre si, seu valor esperado é calculado a partir da média de n amostras (x_i), conforme expresso na equação 11.

$$E(\xi) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{10}$$

$$E(\xi) = \sum_{i=1}^{n} x_i p_i \tag{11}$$

Variáveis aleatórias contínuas

É chamada de variável aleatória contínua (ξ), toda variável que pode assumir um valor x_i, em um intervalo entre dois números quaisquer(a e b), levando em consideração a função de densidade de probabilidade p(x). Se tenho um intervalo arbitrário (a',b'), contido em (a,b), ou seja, $a \leq a', b' \leq b$, em que a probabilidade de ξ é dada pela integral da função de densidade (p(x)) neste intervalo, como o exposto na seguinte equação.

$$P\{a' \leq \xi \leq b'\} = \int_{a'}^{b'} p(x)dx,$$

Como a equação da probabilidade é uma integral em um intervalo (a,b), se a=b, a probabilidade $P\{a' \leq \xi \leq b'\} = 0$, porque a probabilidade de uma função qualquer é a área(A) da região abaixo da função dentro do intervalo em questão, como é ilustrado pela região amarela da figura a seguir (representando a probabilidade da variável aleatória no intervalo (a',b')).

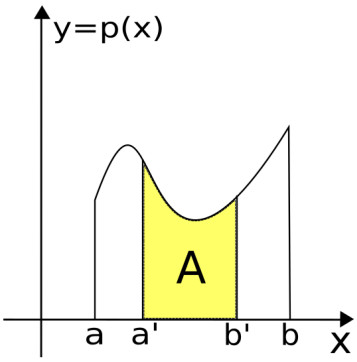


Figura 8: Probabilidade de uma variável aleatória contínua.

Lembrando que a soma de todas as probabilidades deve ser igual a 1, isto quer dizer que, a integral de p(x), no intervalo (a, b), deve ser igual a 1, para garantir a ξ as propriedades básicas de uma variável aleatória e como toda probabilidade deve ter um valor maior ou igual a zero, p(x) deve ser maior ou igual a zero em qualquer intervalo entre a e b.

Valor esperado de uma variável aleatória contínua

O valor esperado de uma variável aleatória contínua, é semelhante à discreta, mas é considerado um somatório de infinitos valores diferenciais, ou seja, uma integral, como descrito na equação a seguir:

$$E(\xi) = \int_{-\infty}^{\infty} xp(x)dx \tag{12}$$

Se utilizarmos a equação 12, e calcularmos a esperança de uma função $f(x)$, obtemos:

$$E(f(x)) = \int f(x)p(x)dx$$

e se calcularmos a esperança de $g(x)$, dado que $g(x) = f(x)/p(x)$, resultando em:

$$E(g(x)) = E(\frac{f(x)}{p(x)}) = \int \frac{f(x)}{p(x)}p(x)dx = \int f(x)dx$$

Embora sejam calculadas de maneiras diferentes, o valor esperado de uma variável aleatória discreta e de uma variável contínua são semelhantes, se for utilizado o maior número de amostras possíveis do intervalo, formando a seguinte relação:

$$E(g(x)) = \int f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} \tag{13}$$

Esta relação chama-se de aproximação Monte Carlo, método muito utilizado para resolver integrais.

Resolvendo a equação de rendering utilizando o método Monte Carlo

Sendo $f(x) = fr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})Li(x, \boldsymbol{\omega_i})\cos\theta_i \ \boldsymbol{d\omega_i}$ a função a ser integrada na equação 7, utilizando o aproximador Monte Carlo (equação 13), obtemos a seguinte aproximação:

$$\int_\Omega fr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})Li(x, \boldsymbol{\omega_i})\cos\theta_i \ \boldsymbol{d\omega_i} \approx \frac{1}{N}\sum_{i=1}^{N}\frac{fr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})Li(x, \boldsymbol{\omega_i})\cos\theta_i \ \boldsymbol{d\omega_i}}{p(x)}, \tag{14}$$

em que p(x) é a minha função de densidade de probabilidade e $fr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})$ é a função de reflectância, que serão discutidas das próximas sessões. O segundo membro da equação é o estimador Monte Carlo para a resolução da integral do primeiro membro (ambas as equações retornam o valor equivalente a Lr da equação 14).

Distribuição de probabilidade

A aproximação da equação de rendering é a mesma para todos os tipos de materiais, a única parte da equação que vai mudar de material para material é a função de reflectância($fr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})$) que define a reflexibilidade do material com relação à luz para cada material e para cada par de direções de incidência e reflexão de luz (ω_i e ω_o). Cada função de BRDF tem uma fr correspondente. Todas as BRDFs podem ser amostradas

com a amostragem uniforme, porém nem todas as BRDFs são amostradas em todas as direções uniformemente, como os materiais difusos (como será visto a nos capítulos a seguir), então como algumas direções importam mais do que outras, existe, na computação gráfica, uma tecnica chamada de amostragem por importância, que calcula direções de reflexão que vão contribuir mais para a cor final do pixel, otimizando o modelo no contexto do algorítmo. Cada tipo de material tem sua forma de distribuir os raios de um modo particular e para cada distribuição escolhida, existe um valor de $p(x)$ correspondente. Para maiores informações sobre amostragem por importância, função de reflectância e distribuição de probabilidade dos tipos mais conhecidos de materiais, e como a amostragem por importância reflete na performance da aplicação, consultar [18] e [25].

A distribuição de probabilidade será calculada de acordo com a amostragem utilizada para calcular as novas direções dos raios de reflexão. Nas sessões de representação dos materiais em *Path Tracing*, serão discutidas as BRDFs e como encaixá-las no integrador Monte Carlo.

2.4 O Algoritmo de *Path Tracing.*

Como discutido anteriormente, a luz se transporta de forma contínua, nas infinitas possíveis direções de emissão e reflexão e se redistribui no ambiente por reflexão até que eventualmente atinja a direção de observação, o que seria impossível de computar em tempo finito. Para simplificar isto, no algorítmo, é calculado apenas um caminho de percorrimento do raio. Contudo, como é muito difícil que esse caminho percorrido pelo raio atinja o interior do sensor, o algorítmo otimiza essa realidade invertendo a ordem de percorrimento do caminho do raio, em que o raio segue o caminho inverso: sai do centro da câmera e atinge os objetos da cena, em que, eventualmente o raio atinja um material emissivo, fazendo com que o sensor tenha acesso ao valor de radiância do primeiro ponto atingido através da recursão. Esta inversão do caminho percorrido pela luz, que otimiza o algoritmo, é ilustrado na figura 9. Além disso, para sintetizar uma imagem de uma resolução MxN, é necessário, no mínimo, disparar MxN raios por *pixel* num plano conhecido na computação gráfica como *view plane*, representado na figura 10.

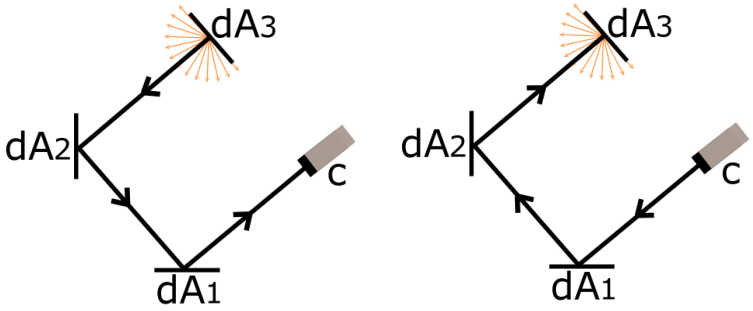


Figura 9: Esboço do caminho percorrido por um raio de luz (como ocorre na natureza e o caminho inverso), onde dA_3 é uma área diferencial de uma fonte de luz, enquanto as áreas diferenciais dA_1 e dA_2 são superfícies que refletem a luz vinda da fonte de luz até atingir o sensor, respresentado na figura como a câmera (c).).

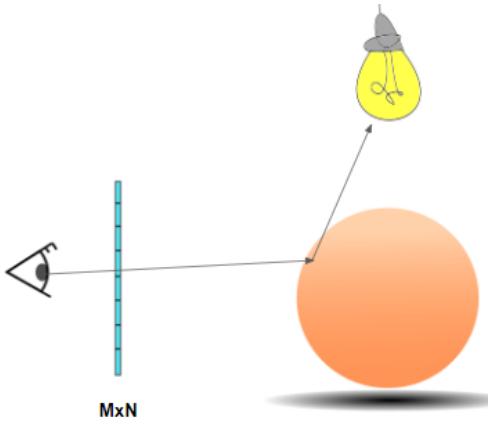


Figura 10: *Plano de visão (representado na figura na cor azul).*

O Plano de visão é dividido em *pixels*, que são as unidades fundamentais de uma imagem. Quanto maior o número de *pixels*, melhor a resolução da imagem final. Para formar a imagem, o algoritmo precisa traçar um raio que saia da posição do sensor, atinja o centro do *pixel* e percorra a cena, atingindo superfícies que brilham por reflexão, ou por emissão. Mas, para melhorar a aproximação Monte Carlo e o valor esperado ($E(\xi)$) do estimador, é necessário que seja feito o maior número de amostras possíveis para cada *pixel* do plano de visão (o ideal é que sejam feitas infinitas amostras para que o valor esperado se aproxime do seu valor ótimo). Para que cada amostra contribua de forma diferente para o valor esperado, é importante utilizar uma boa função de distribuição, para que as amostras sejam feitas dentro de área de cada *pixel* do *view plane* de maneira mais distribuída possível, cobrindo a maior área possível da área total. Para encontrar maiores informações sobre o caminho de percorrimento do raio de luz, o funcionamento do *Path Tracing* e as melhores maneiras de distribuir as amostras de raios em cada pixel, consultar [5] e [6]. Além de tudo isto, cada raio lançado pelas amostras de cada *pixel* deve

fazer um cálculo de teste de intercecção com todos os triângulos da geometria da cena a fim de verificar qual o ponto mais próximo que um raio qualquer atingiu ao percorrer a cena inteira. Todas estas questões colaboram com a alta demanda de processamento de dados do algorítmo.

2.5 A eficiência do algoritmo de *Path Tracing*.

Como podemos ver, os principais problemas do algoritmo relacionam-se à alta demanda de custo computacional das aplicações em decorrência do número de operações. Para que seja calculado o valor ótimo do valor esperado ($E(\xi)$) de radiância (Lr) que reflete de um objeto da cena e atinge o sensor, é necessário um grande número de amostras de radiância refletida. Além de ser necessário disparar o máximo possível de raios por cada pixel, para calcular os valores de irradiância vindos de cada ponto atingidos por cada raio (para isto, são calculadas as irradiâncias (E) e suas radiâncias refletidas (Lr), são calculadas a partir das irradiâncias, que por sua vez são calculadas à partir das infinitas radiâncias incidentes (Li), multiplicando este valor calculado de radiância pela BRDF e dividindo pela funcão de densidade de probabilidade, somando todos os valores e resultando na irradiância). As chamadas que disparam estes raios de luz na cena são chamadas recursivas que ocorrem para cada *pixel* de uma tela de resolução MxN, fazendo com que se repitam estes cálculos MxN vezes e ainda multiplicado por um valor K de número de amostras.

Para atingir o valor ótimo, alto custo computacional é exigido devido o número de chamadas recursivas para que este valor de radiância seja calculado para cada *pixel*. A aproximação Monte Carlo vem a facilitar, transformando um somatório de infinitas parcelas diferenciais e contínuas numa soma finita de valores discretos, que são nossas amostras(quanto maior o número de amostras, mais próximo do valor exato), reduzindo, mas não viabilizando, o custo computacional para calcular o valor aproximado do exato utilizando o mesmo algorítmo, como expresso pela equação 15.

Para reduzir o custo computacional, são definidos valores fixos de número de amostras por pixel, ou seja, quantos raios serão disparados por pixel. É necessário também a definição de bons critérios de parada para as chamadas recursivas (que geralmente é definida pelo número de chamadas dos raios secundários, podendo ser um valor fixo escolhido pelo usuário, ou a parada da recursão pode ocorrer em consequência da colisão do raio com uma fonte de luz.

Para reduzir ainda mais o custo do algoritmo, são desconsideradas as infinitas radiâncias incidentes, uma vez que um raio atinge uma superfície, seriam necessários lançar infinitos raios secundários em todas as direções para que se calculasse a radiância correspondente a energia refletida na direção do pixel, o que é otimizado com através de

um maior número de raios sendo lançados por pixel, o que acaba substituindo o somatório e a média das amostras do estimador Monte Carlo. Esta simplificação reduz a equação 14 da página 27 da seguinte forma:

$$Lr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{fr(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) Li(x, \boldsymbol{\omega_i}) \cos\theta_i \ \boldsymbol{d\omega_i}}{p(x)} \tag{15}$$

Esta equação reduzida simplifica a equação de rendering e sua implementação. A seguir, encontra-se um pseudo-código do algorítmo do *Path Tracing*, para melhor compreensão do seu funcionamento.

```
Color TracePath(Ray ray, count depth) {
    if (depth >= MaxDepth) {
      return Black;  // Condicao de parada quando atinge numero maximo pre-
    definido de chamadas recursivas (MaxDepth).
    }

    ray.EncontraObjetoMaisProximo(); //na direcao do raio
    if (ray.Atingiu == false) {
      return Preto;  // Nao atingiu nenhum objeto da cena (cor de fundo da
    imagem).
    }

    Material material = ray.thingHit->material;
    Color Ke = material.emitancia;

    // Calcla direcao aleatoria para proximo raio do caminho.
    Ray newRay;
    newRay.origin = ray.PontoDeInterseccao;

    newRay.direction = Amostragem(ray.NormalDaSuperficie);

    // funcao de distribuicao de probabilidade do aproximador
    const float pdf = material.pdf_aproximador;

    // Compute the BRDF for this ray
    float cos_theta = DotProduct(newRay.direction, ray.NormalDaSuperficie);
    Color BRDF = material.brdf();

    // Caminho recursivo dos raios secundarios.
    Color Li = TracePath(newRay, depth + 1);

    // Calcula a euqacao de rendering.
    return Ke + (BRDF * Ki * cos_theta / pdf);
  }


```

```
35  void Render(Image finalImage, count numSamples) {
36    foreach ( pixel in finalImage) {
37      foreach (i in numSamples) {
38        Ray r = camera.generateRay(pixel);
39        pixel.color += TracePath(r, 0);
40      }
41      pixel.color /= numSamples;  // Media das amostras.
42    }
43  }
```

Além destas otimizações, existem, na literatura, soluções para o problema dos testes de intercecção que aceleram o tempo do teste de intercecção de cada raio com todos os triângulos da geometria, através de estruturas de dados de aceleração, como por exemplo as BVHs [13], que testam apenas os triângulos contidos dentro dos volumes, organizado em hierarquias. Para o problema das amostras demorarem a convergir para o valor ótimo, existe uma técnica de aceleração, desenvolvida para gerar amostras que colaborem mais para o valor esperado ser atingido com menos tempo, lançando raios em direção à fonte de luz, a fim de encontrar valores maiores do que zero de radiância incidente nos pontos mais rápido, melhorando a qualidade das amostras coletadas, chamada de amostragem direta da luz [34].

Mesmo com todas estas otimizações, com o auxílio do estimador Monte Carlo e todas as aproximações numéricas utilizadas, o algoritmo continua demandando alto custo computacional, principalmente quando se trata de aplicações em tempo real. Uma das alternativas de otimizar o algoritmo e seu funcionamento em tempo real é fazendo uso da programação paralela, que é viabilizada, principalmente, através da robustez do *Hardware* da GPU e seus múltiplos núcleos de processamento, como se explicará imediatamente, na sessão a seguir.

2.6 A eficiência do *Path Tracing* na GPU.

Matthew Scarpino, em [36], no primeiro capítulo, traz uma discussão do quanto a GPU (Unidade de processamento gráfico da placa de vídeo) aprimorou a performance das aplicações. Por ter uma arquitetura diferente da CPU, com mais unidades de processamento, com memória local para cada unidade e oferecer, em *hardware*, suporte para instruções vetoriais, processamento paralelo de dados, leitura e escrita rápidas de *buffers* de dados, resumindo, suporte para pipeline gráfico em sua totalidade, a GPU tornarou-se uma excelente opção para a programação de qualquer pipeline gráfico, por favorecer práticas de programação paralela e por suportar todas as operações algébricas básicas entre vetores e operações aritméticas básicas no geral.

No mesmo livro, o autor explica o funcionamento da arquitetura da GPU, a fim

de tirar o máximo de proveito do *hardware* com o objetivo de melhorar a performance das aplicações. As instruções vetoriais são um dos maiores inimigos na performance das aplicações, porque cada vetor tem um númerdo de componentes e para cada operação, seja ela de soma, ou subtração, deve ser realizada componente a componente, um de cada vez. No algoritmo do *Path Tracing*, muitos dados são representados por vetores, como as cores, os raios, os índices de refração e as constantes dos materiais, e assim por diante. Com o suporte às instruções vetoriais, a GPU torna-se uma grande aliada na programação de renderizadores, em especial o algoritmo escolhido no nosso estudo, visto que a GPU possui, em *hardware*, suporte às operações de cálculo vetorial, realizando as mesmas operações de soma para um inteiro, como também para um vetor de inteiros de três posições, num mesmo tempo, reduzindo o custo computacional destas operações, em tempo real, comparando com as aplicações feitas em CPU. Além disso, possui um número maior de núcleos de processamento, o que aumenta o poder de processamento paralelo de dados, favorecendo particionamento de dados para agilizar as chamadas das rotinas das funções do renderer para cada *pixel* do *display* da imagem de alta resolução. Por isso, a GPU é um dos artifícios utilizados na literatura para melhorar a eficiência do algorítmo do *Path Tracing*.

Para informações mais detalhadas na influência da arquitetura da GPU na melhoria da performance das aplicações na computação gráfica, devido à demanda computacional dos renderizadores e o suporte em *hardware* que a GPU oferece para as operações fundamentais dos mesmos, consultar [40].

3 REPRESENTAÇÃO DE MATERIAIS EM *PATH TRACING*

Na síntese de imagens fotorrealistas, no contexto do algorítmo em GPU, o que há de fundamental para obter resultados em que não haja distinção de uma fotografia para uma imagem renderizada (sintetizada pelo computador) é programar modelos físicos e matemáticos que simulem e descrevam os diversos tipos de materiais e suas propriedades particulares na interação com a luz na natureza para replicá-los digitalmente (sua transparência, seu grau de pureza, se o tipo de material é hetero ou homogêneo, sua reflexibilidade, entre outras características [23]), ou modelos *empíricos* [18], que embora não simulem os fenômenos físicos da luz, como o transporte de energia luminosa, e por isso não respeitem as leis de conservação de energia, alcancem bons resultados, ou resultados aproximados aos modelos fisicamente plausíveis. Segundo Naty Hoffman [23], para que a BRDF seja considerada um modelo físico, ou físicamente plausível, é necessário respeitar as leis de conservação de energia e reciprocidade da função com respeito à ordem dos parâmetros. Para que a BRDF conserve energia, nunca poderá retornar um valor de energia superior a quantia de energia luminosa que incidiu no ponto, nem uma quantia negativa (ou seja, um valor positivo e que a integral da BRDF deve retonrar um valor entre zero e um). Para que ela seja recíproca, é necessário que $fr(\boldsymbol{\omega_i} \leftrightarrow \boldsymbol{\omega_o}) = fr(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = fr(\boldsymbol{\omega_o}, \boldsymbol{\omega_i})$, ou seja, a ordem dos parâmetros não altere o valor retornado pela função. Caso a BRDF não respeite alguma dessas propriedades, ela não é considerada físicamente plausível.

A função de distribuição de reflectância bidirecional é descrita por equações matemáticas (recebendo os ângulos sólidos como parâmetro), mas também pode ser representada por um banco de dados- como em [22]-, em que para cada direção $\boldsymbol{\omega_i}$, relaciona-se uma proporção correspondente à direção $\boldsymbol{\omega_o}$. Em outras palavras, para cada par de ângulo sólido ($\boldsymbol{\omega_i}$,$\boldsymbol{\omega_o}$), temos um valor de proporção salvo na memória (banco de dados), causando uma certa imprecisão, a medida que um intervalo infinito é discretizado em poucos valores de posição no espaço 3D), inviabilizando o uso desse método para videogames, pois o mercado preza por produtos de boa qualidade e baixo custo de memória e de processamento de dados [41].

O algorítmo recursivo do *Path Tracing* recebe como um de seus parâmetros um raio, que contém uma origem e uma direção. A medida que o raio percorre a cena, partindo da origem, seguindo a direção do primeiro raio (mapeados randômicamente em direção a um ponto da região ao redor do centro do pixel), e caso cruze um objeto da cena, o ponto de cruzamento será a nova origem e a nova direção do próximo raio e seu sentido é definido pela direção de reflexão do ângulo sólido de saída, ou reflexão, que é calculado pela amostragem uniforme (explicada mais adiante) , por exemplo. Porém para simplificar e compatibilizar as direções dos ângulos sólidos, com as direções dos raios, é

utilizada a notação de vetor para representar as direções dos ângulos sólidos com vetores de três coordenadas (x,y,z).

Para resolver localmente a direção de reflexão são utilizadas coordenadas esféricas, para isso, é necessário mudar apenas a direção e o sentido do raio incidente e assim formar o raio refletido, pois, por fazer uso das variáveis deste sistemas de coordenadas (θ e φ), faz com que haja a necessidade de inverter a direção do reaio de incidência para poder representá-los nesta notação. A forma de representar os vetores é modificada para coordenadas polares por razões de performance. Na figura a seguir, está ilustrada a maneira que parametriza-se um vetor qualquer ($\boldsymbol{v}$) em coordenadas esféricas, representados no sistema local do ponto de intercecção da superfície em coordenadas esféricas.

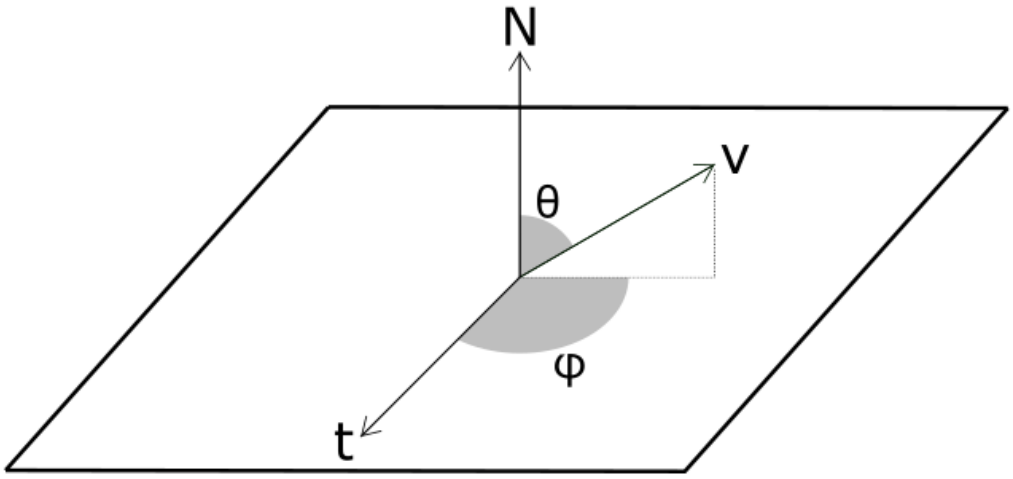


Figura 11: Coordenadas esféricas. $\boldsymbol{V}$ é um vetor que representa uma direção qualquer, $\boldsymbol{t}$ um vetor paralelo à superfície e $\boldsymbol{N}$ o vetor normal da superfície.

Para maiores detalhes em coordenadas esféricas, verificar [5].

Com esta mudança de sistema de coordenadas para o sistema local de ângulo sólido para coordenadas esféricas, a função de reflectância é denotada por $fr(\theta_i, \varphi_i, \theta_o, \varphi_o)$. Por motivos de simplificação, ou de coesão matemática, em alguns momentos a notação antiga será utilizada, recebendo como parâmetros ângulos sólidos de incidência e de reflexão, ou a nova, recebendo os ângulos das coordenadas esféricas de cada direção (incidência e reflexão).

3.1 Como se classificam os modelos de BRDF que serão estudados.

Para que possamos perceber visualmente os objetos e as coisas em nossa volta, é necessário que existam objetos que *emitam*, *transmitam*, ou *retransmitam* a luz na direção dos nossos olhos de maneiras diversificadas, representando a diversidade dos materiais existentes. Como foi dito na subsessão anterior, é possível descrever materiais em *Path*

Tracing utilizando modelos de BRDF que podem ser baseados em modelos físicos, como também podemos descrever os materiais em *Path Tracing* utilizando modelos *empíricos*, que chegam em resultados semelhantes e aproximados, porém não são baseado em física e, podem não conservar energia, e dependendo da aplicação, não serem úteis, mas como estamos avaliando a plausabilidade dos modelos em *Path Tracing* no contexto de desenvolvimento de *renderers* em tempo real na GPU e o objetivo do estudo é, também, analisar aproximações que reduzam um pouco o realismo, desde que melhore a performance da aplicação, com a finalidade de comparar seus resultados com os modelos baseados em física, que geralmente são mais caros computacionalmente e, desta forma, analisar o custo-benefício de cada um. Existem diversos modelos empíricos e físicamente plausíveis, para os diversos tipos de materiais. Estudaremos dois tipos de materiais: Difuso e Especular, por serem algorítmos conhecidos e amplamente utilizados no contexto do nosso estudo, dentre os quais selecionamos quatro modelos a serem estudados os quais serão detalhados adiante. Além destes quatro modelos, existem outros diversos modelos de BRDF na literatura utilizados em *Path Tracing*, como, por exemplo, o modelo difuso empírico, para simular o comportamento difuso da lua, que se encontra em [33], dois outros modelos difusos fisicamente plausíveis, que se encontram em [37] e [38] e modelos especulares anisotrópicos físicamente plausíveis, como em [30], entre outros modelos discutidos e utilizados na computação gráfica. Os modelos que serão estudados estão devidamente classificados segundo tipo de material e plausabilidade do modelo na figura 12.

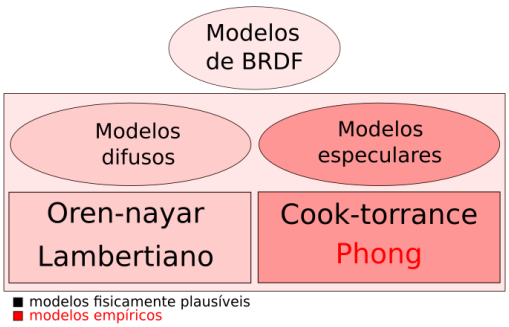


Figura 12: Tabela de modelos de funções de distribuição de reflectância bidirecional.

Este modo de classificar os modelos e as diversas funções de distribuição de reflectância, expostos na figura 12, foi baseado no modelo de classificação detalhado em [18].

Materiais com brilho difuso, são objetos que existem na natureza, que não são meios *transmissíveis* para a luz, isto significa que a luz não atravessa esse determinado tipo de material, onde parte da energia é absorvida e outra parte é refletida. Lembrando

que em materiais difusos, não há reflexão direta da luz em uma única direção, ou ao redor de uma direção (como em espelhos e plásticos), o que significa que em materiais difusos, não há nenhuma forma de efeito de reflexo como podemos ver nosso reflexo no espelho e superfícies metálicas polidas, à exemplo do concreto. Na natureza, os materiais difusos distribuem a energia de maneira quase que uniforme em todas as direções (esta uniformidade varia de acordo com o grau de rugosidade da superfície difusa do material), enquanto nos modelos especulares, a maior parte da energia refletida é mais concentrada numa determinada região, o que resulta num brilho especular.

Neste documento, será avaliada a performance de dois modelos difusos: o modelo da BRDF perfeitamente difusa, também chamada de *lambertiana*, proposto por JH. Lambert [20], e o modelo de *Oran-Nayar*, proposto por Michael Oren e Shree K. Nayar [26].

Modelo de Lambert.

Neste modelo, considera-se que a superfície é *optciamente* planar (ou seja, não possui irregularidades nem em escalas de diferencial de área), mas também que a contribuição de energia da irradiância incidente, contibui com um valor constante de radiância refletida em todas as direções (e vise-versa, dada a comutatividade da função de distribuição bidirecional de reflectância). Partindo deste pressuposto, podemos representar esse modelo físico da seguinte forma:

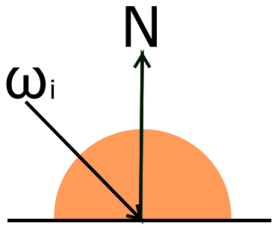


Figura 13: Modelo de BRDF *lambertiana*.

Sendo ω_i o ângulo sólido de incidência e o semi-círculo laranja, a distribuição da energia incidente nas infinitas possíveis direções de reflexão. Na natureza, nenhum material distribui energia unifomemente, de maneira igual em todas as direções, porém, materiais como a areia distribuem a energia de forma semelhante, a depender do grau de rugosidade da superfície, sendo perfeitamente polida, reflitiria luz desta maneira.

Como é descrito em [5], a função de reflectância difusa, para manter-se conservando energia e distribuir os raios uniformemente em todas as direções, é descrita da seguinte forma:

$$fr_{Lambert}(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{\rho(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})}{\pi} \tag{16}$$

Onde $\rho(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})$ é uma constante do modelo de Lambert chamada, na literatura [26], de albedo e x é um ponto qualquer da superfície de um objeto qualquer.

Com esta função de reflectância, a equação de rendering, para este modelo, é calculada da seguinte maneira:

$$Lo = Le(x, \boldsymbol{\omega_o}) + \int_{\Omega} \frac{\rho(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o})}{\pi} Li(x, \boldsymbol{\omega_i}) \cos\theta_i \ \boldsymbol{d\omega_i} \tag{17}$$

Como o valor da PDF do aproximador Monte Carlo para a amostragem uniforme é igual a $\frac{1}{2\pi}$, temos:

$$Lo \approx Le(x, \boldsymbol{\omega_o}) + \frac{1}{N} \sum_{i=1}^{N} \frac{\rho(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) Li(x, \boldsymbol{\omega_i}) \cos\theta_i \ \boldsymbol{d\omega_i}}{\pi/2\pi}, \tag{18}$$

o que resultaria na equação a seguir:

$$Lo \approx Le(x, \boldsymbol{\omega_o}) + \frac{2}{N} \sum_{i=1}^{N} \rho(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) Li(x, \boldsymbol{\omega_i}) \cos\theta_i \ \boldsymbol{d\omega_i} \tag{19}$$

Modelo de Oren-Nayar

Como dito em [17], pouca atenção é dada aos modelos fisicamente plausíveis difusos, isto acontece porque o modelo de Lambert é uma solução de baixo custo de processamento, já que o seu valor permanece constante independente da direção de reflexão e alcança resultados considerados razoáveis para as diversas aplicabilidades do *rendering* baseado em física. Em contrapartida, foi proposto em 1994, por Shree K. Nayar e Michael Oren um modelo difuso baseado em microfaces [26]. A modelagem das microfaces é baseado no modelo de *Torrance-Sparrow* [24], onde as microfaces são interpretadas como cavidades-V, que o modelo de *Cook-Torrance* utiliza.

A equação do modelo de *Oren-Nayar* é dada pela equação 20, como em [8].

$$fr_{Oren-Nayar}(\boldsymbol{h}, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{1}{|\boldsymbol{N} \cdot \boldsymbol{s}||\boldsymbol{N} \cdot \boldsymbol{\omega_o}|} \int_{MG} (\boldsymbol{h} \cdot \boldsymbol{s})(\boldsymbol{h} \cdot \boldsymbol{\omega_o}) F_{diff}(\boldsymbol{\omega_i}, \boldsymbol{h}) G(\boldsymbol{s}, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{h}) D(\boldsymbol{h}) d\omega_h \tag{20}$$

Onde MG é a superfície de toda a *microgeometria* em cavidades-V e $\boldsymbol{s}$ é a direção da fonte de luz (em direção à micro-superfície).

Como podemos observar, a equação do modelo de *Oren-Nayar* é bem diferente da equação do outro modelo difuso, estudado na sessão anterior, apresenteando outros

termos, como o termo geométrico (G), em algumas literaturas (como em [26]) chamado de termo de atenuação geométrico, a função de reflectância, neste caso difusa (F_{diff}), e a função de distribuição das normais (D). Para compreendermos a equação deste modelo, se faz necessário a explanação de algumas características peculiares dos materiais difusos rugosos, modelados em cavidades-V, como se demonstra nas figuras 14 e 15.

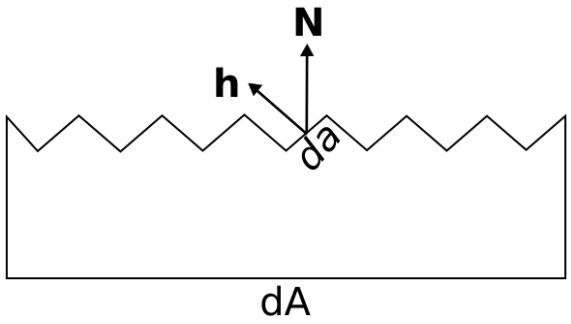


Figura 14: Modelo de microfaces baseado em cavidades-V (modelo isotrópico, modelo do nosso estudo).

Neste modelo, a microgeometria é composta por micro-áreas de mesma área (da) e de mesma orientação ($\boldsymbol{h}$ e $\boldsymbol{h} + \pi$), onde $\boldsymbol{h}$ é a normal da micro-superfície, como o visto na figura 14, distribuidos uniformemente em orientação ao plano da macro-superfície dA. Para que a compreensão fique intuitiva, é como se o modelo das cavidades-V, modelando materiais difusos isotrópicos, transformasse um ponto, ou diferencial de área dA, numa colessão de áreas ainda menores, da, em que estas superfícies não são como no modelo de microfaces especular, pois são superfícies difusas. Esta colessão de micro-superfícies, compõe a *microgreometria* de cada diferencial de área do modelo.

Quanto à atenuação do brilho por mascaramento e sombreamento [26], como também a moderação desses efeitos em consequência da iluminação global dentro da microgeometria, é inclusa neste modelo uma função denominada, na literatura, fator de atenuação geométrica (G), este termo é baseado no mesmo termo geométrico da equação do modelo de *Torrance-Sparrow* [24], que utiliza uma abordagem de microfaces baseado nas cavidades-V. As figuras a seguir ilustram estes efeitos de atenuação de brilho nas microfaces.

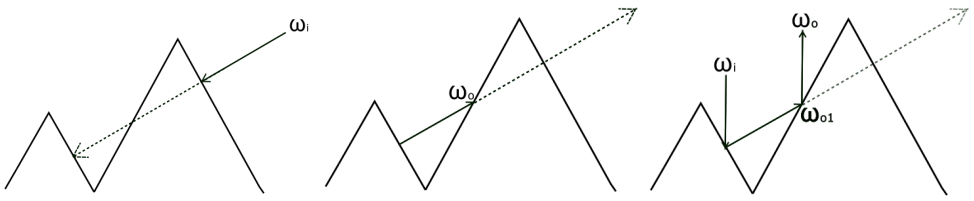


Figura 15: Sombreamento, mascaramento e múltiplas reflexões entre as microfaces.

Na primeira imagem, da figura 15, todas as micro-superfícies poderiam ser atingidas pelo raio de luz incidente ($\boldsymbol{\omega_i}$), porém, a superfície mais à direita foi atingida primeiro, deixando as outras superfícies em sombra, para essa direção de incidência. Na segunda imagem, da mesma figura, a direção de observação, partindo do ponto de partida da mesma, é escondida por outra micro-superfície, ou seja, esta micro-superfície de onde a direção de observação ($\boldsymbol{\omega_o}$) partiu, foi mascarada pelas outras micro-superfícies. Na terceira imagem, encontra-se um exemplo em que a direção de observação ($\boldsymbol{\omega_o}$) foi modificada e recebe colaboração energética decorrente de multiplas reflexões do raio incidente ($\boldsymbol{\omega_i}$), simulando um efeito de iluminação global de multiplas reflexões na microgeometria. O termo geométrico, que simula estes efeitos, é dado por:

$$G = Min\left[1, Max\left[0, \frac{2(\boldsymbol{s}\cdot\boldsymbol{N})(\boldsymbol{h}\cdot\boldsymbol{N})}{(\boldsymbol{s}\cdot\boldsymbol{h})}, \frac{2(\boldsymbol{\omega_o}\cdot\boldsymbol{N})(\boldsymbol{h}\cdot\boldsymbol{N})}{(\boldsymbol{\omega_o}\cdot\boldsymbol{h})}\right]\right]$$

As micro-superfícies são consideradas superfícies opticamente planas, porém difusas, com suas dimensões superiores ao comprimento de onda da luz, o que quer dizer que a luz reflete na micro-superfície difusa com proporção da reflectância semelhante ao modelo de Lambert. O termo que retorna a proporção da BRDF correspondente ao percentual dos raios que são refletidos vindo das micro-superfícies da micro-geometria cuja normal ($\boldsymbol{h}$) coincide com a normal da macro-superfície ($\boldsymbol{N}$), é o termo D, dado pela seguinte equação:

$$D = c \times e^{-\frac{\theta_a^2}{2\sigma^2}}$$

O termo correspondente a reflectância difusa (F_{diff}) é dado por:

$$F_{diff}(\boldsymbol{l}, \boldsymbol{h}) = (\boldsymbol{N}\cdot\boldsymbol{\omega_i})\frac{\rho}{\pi}$$

Para concluir, os termos restantes da equação 20, os produtos internos, servem para simular um efeito especial dos materiais difusos rugosos, chamado de *backscattering*, que será explicado imediatamente.

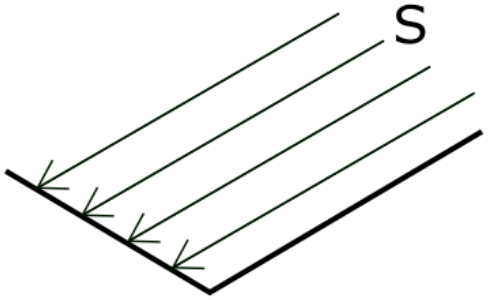


Figura 16: Modelagem da cavidade-V: Incidência de luz direto da fonte de luz (direção e sentido do vetor S).

Considere uma cavidade-V difusa, composta por um par de micro-superfícies planares de área *da*, com orientações concorrentes e simétricas entre si. Observando a figura 16, que é atingida por raios de luz provenientes de uma fonte de luz na direção $\boldsymbol{s}$, a micro-superfície do lado esquerdo recebe uma contribuição energética da fonte de luz maior do que a micro-superfície do lado direito. Desta forma, a micro-face esquerda brilha mais do que a direita, contudo existem materiais difusos como os que compõe a superfície da lua, como também o gesso e a areia, que apresentam um comportamento variado, a depender da direção de observação ($\boldsymbol{e_1}$ e $\boldsymbol{e_2}$), como mostram as figuras a seguir.

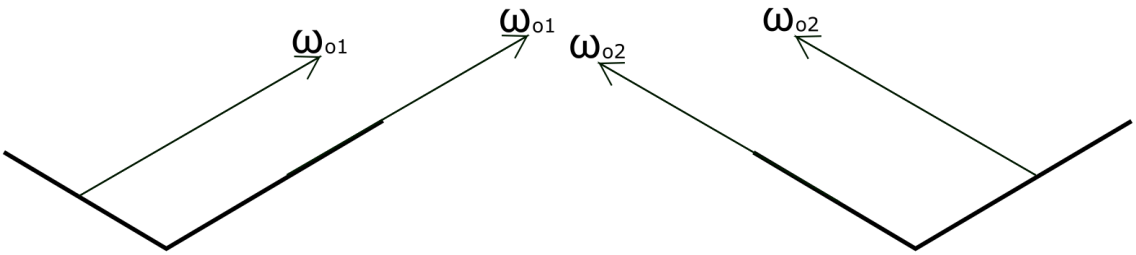


Figura 17: Modelagem da cavidade-V: Direção de observação.

Em decorrência do *backscattering*, levando em consideração a direção $\boldsymbol{s}$ da figura 16, se a direção de observação for $\boldsymbol{\omega_{o2}}$, resultará num escurecimento do brilho, decorrente a redução da reflectância especular difusa, em proporção ao desvio da direção de observação aumenta em relação ao inverso da direção da insidência. Ou seja, nos casos em que a direção de observação coincidir exatamente com a direção do inverso da direção de incidência da luz vinda direto da fonte de luz, $\boldsymbol{\omega_{o1}}$, a reflectância atinge seu valor máximo (se a fonte de luz for pontual e única fonte de luz da cena, não havendo contribuição de outros pontos de luz), causando uma inconstância na distribuição de energia ao redor do ponto de incidência luminosa num objeto difuso, divergindo do modelo difuso de Lambert, por não distribuir os raios de uma forma igual em todas as direções, independente da direção de observação, apresentando uma vantagem do modelo de Oren-Nayar em comparação com o modelo difuso de Lambert, pois, apesar do modelo de Lambert apresentar baixo custo de processamento, foi visto em outras referências, [27], [26] e [28], que o modelo lambertiano é uma aproximação muito imprecisa, ou inadequada, para simular alguns efeitos da reflectância difusa como o observado em materiais como o pano, areia, argila e gesso e em [15], onde foi observado o comportamento da reflectância difusa da lua e o quanto é incompatível do que é visto no modelo de reflectância difusa de Lambert, uma vez que em superfícies difusas como estas, a luz tende a refletir uma maior parcela da energia incidente na direção da fonte de luz (*retrorreflexão* [18], ou *backscattering* [27]).

Como a equação 20 é uma equação grande e complexa (inviáveis para aplicações em tempo real), em [26], foram propostas duas aproximações: a primeira, descrita por equações maiores, responsáveis por simular os fenômenos de interreflexão. A segunda

aproximação, descrita como se segue na equação 21, não simula interreflexão, o que, segundo [26], se dá pela ausência dos termos que predominam na primeira aproximação, que demandam alta performance da aplicação e fazem pouca diferença no brilho do resultado final (o que motivou a utilização destas equações nos nossos testes).

$$fr_{Oren-Nayar}(\boldsymbol{\omega_o}, \boldsymbol{\omega_i}) = \frac{\rho}{\pi}(A + B\ Max(0, \cos(\phi_{\boldsymbol{\omega_i}} - \phi_{\boldsymbol{\omega_o}}))\sin(a)\tan(b)) \qquad (21)$$

Onde $a = max(\theta_{\boldsymbol{\omega_o}}, \theta_{\boldsymbol{\omega_i}})$, $b = min(\theta_{\boldsymbol{\omega_o}}, \theta_{\boldsymbol{\omega_i}})$, $A = 1 - 0.5\frac{\alpha_m^2}{\alpha_m^2 + 0.33}$, $B = 0.45\frac{\alpha_m^2}{\alpha_m^2 + 0.09}$ e os termos $\theta_{\boldsymbol{\omega_i}}$ e $\phi_{\boldsymbol{\omega_i}}$ são as coordenadas esféricas do vetor $\boldsymbol{\omega_i}$ e, do mesmo modo, $\theta_{\boldsymbol{\omega_o}}$ e $\phi_{\boldsymbol{\omega_o}}$ são as coordenadas esféricas do vetor $\boldsymbol{\omega_o}$. O termo α_m, por sua vez, é um número real entre zero e um que define quão áspera é a superfície do material. Quanto maior a rugosidade, maior o efeito de retrorreflexão, como explícito em [18]. É importante notar que esta aproximação, quando α é igual a zero, a equação é reduzida para a equação do modelo de Lambert($\frac{\rho}{\pi}$).

O que há de mais interessante no modelo é capacidade de simular o comportamento do brilho da lua e se assemelhar aos materiais difusos que existem na natureza. O termo responsável por este brilho mais forte nas bordas das quádricas implementadas com este modelo é o termo especular, descrito em [27], é uma equação semelhante ao termo D descrito nesta mesma sessão.

Como a segunda aproximação apresentada em [27] e [29] (equação 21) é mais utilizada na literatura da computação gráfica (como em [18]), além disso, abrir mão da simulação destes efeitos, através desta aproximação apresentada, se torna necessário devido a inviabilidade da integração da equação inteira em aplicações em tempo real, como explicado no próprio artigo em que as equações e suas aproximações foram apresentadas [27] [26]. Lembrando também que o objetivo do nosso trabalho é analisar a semelhança dos modelos e aprimorar a performance. Por estas razões, tornou-se mais conveniente utilizar a aproximação conforme descrito pela equação 21.

Assumindo que a amostragem dos raios secundários deste modelo é uniforme e, por isso, sua PDF ($p(x)$) da aproximação Monte Carlo é igual a $\frac{1}{\pi}$, o modelo de *Oren-Nayar* pode ser aplicado na equação de rendering da seguinte forma:

$$Lo \approx Le(x, \boldsymbol{\omega_o}) + \frac{2}{N}\sum_{i=1}^{N} \rho(A + B\ Max(0, \cos(\phi_{\boldsymbol{\omega_i}} - \phi_{\boldsymbol{\omega_o}}))\sin(a)\tan(b)) Li(x, \boldsymbol{\omega_i})\cos\theta_i\ \boldsymbol{d\omega_i} \qquad (22)$$

Materiais especulares, por outro lado, a depender do grau de transmissividade do material, podem ser transparentes, translúcidos ou opacos. Além da transmissividade, os materiais com brilho especular, possuem a propriedade de refletir a luz de maneira

discontínua, ou refletir de modo mais concentrado ao redor de uma determinada direção, dependendo do seu grau de *rugosidade* da superfície do material. Como visto em [23], para simular materiais com brilho e reflexão especulares, é necessário simular estes efeitos, dentre eles, estudaremos os que se encontram a seguir:

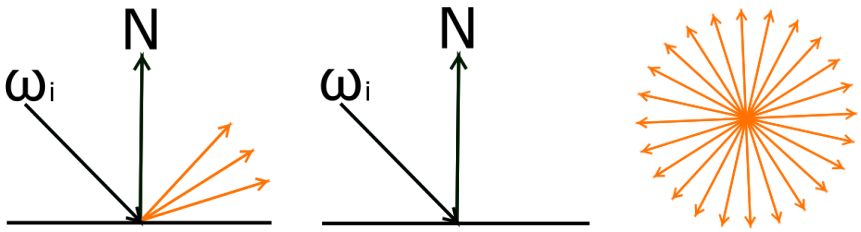


Figura 18: Possíveis formas da luz interagir com o meio especular opaco (reflexão, absorção e emissão).

Onde $\boldsymbol{N}$ é o vetor normal da superfície em questão e $\boldsymbol{\omega_i}$ é o vetor que representa a direção de incidência de luz. Os vetores desenhados em laranja na primeira imagem da figua 18 são as direções de reflexão, enquanto na terceira, são as direções de emissão de luz. Esta figura ilustra, respectivamente a reflexão, a incidência e a emissão.

Cada "raio"de luz carrega uma certa quantidade de energia espectral que conhecemos pelo nome de *radiância*, essa quantidade de energia espectral pode assumir diversos valores num intervalo contínuo e infinito de comprimento de onda, também estudado nas sessões anteriores, onde nos interessa apenas a faixa visível de comprimento de onda da luz (aproximadamente entre 400 e 780 nm), mas o feixe de luz, seja ela visível ou invisível, interage com o meios de brilho especular da mesma forma: *reflexão, espalhamento, absorção* e *emissão*.

No nosso estudo, trataremos dos modelos que simulam apenas a *reflexão* e a *absorção*, pois serão retratados apenas modelos especulares opacos. A reflexão ocorroe quando a energia espectral da radiância incidente é convertida em energia espectral em qualquer direção. A *absorção*, representada na terceira imagem da figura 18, ocorre em materiais metálicos como o cobre, em que, quando a luz não é reletida, é transmitida e imediatamente absorvida pelo meio. Estudaremos dois modelos *isotrópicos* dos materiais especulares: o modelo de *Phong*, proposto por Bui Tuong Phong [29] (classificado como um modelo empírico em [18]) e o modelo de *cook-torrance*, proposto por Robert L. Cook e Kenneth E. Torrance [31], que é um modelo fisicamente plausível, que será detalhado adiante.

Modelo de Phong.

O Modelo proposto por Bui Tuong Phong, em [29], é um modelo criado para se adequar a outro *pipeline* de renderização de imagens, ou seja, outro método de sintetização de imagens diferente do *Path Tracing*: O *pipeline* de *rasterização* não é baseado em física e, portanto, busca se assemelhar ao comportamento real dos objetos através de cálculos numéricos que chegam a resultados considerados semelhantes ao que existe na vida real. Por isto, este modelo é considerado *empírico* e, em consequência disso, não é um modelo fisicamente plausível.

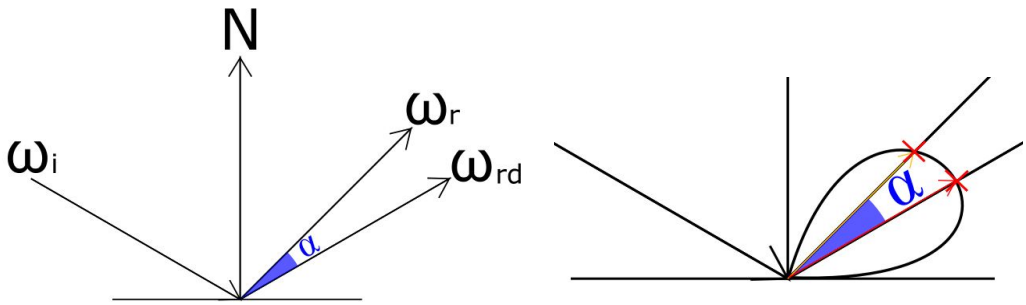


Figura 19: Modelo de BRDF de Phong.

No caso deste modelo, a energia não é refletida de maneira uniforme em todas as direções, mas concentradas em torno da direção de reflexão, variando a largura do sólido de reflexão de acordo com o parâmetro α que representa sua rugosidade. Este sólido é representado na figura 19 por um desenho de formato semelhante a uma gota perfeitamente simétrica, onde a maior parcela da energia refletida, encontra-se na direção de reflexão e o restante distribuido ao seu redor, em menor intensidade. Em decorrência disso, como explícito em [29], a equação do modelo de Phong é descrita da maneira a seguir:

$$fr_{Phong}(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = C_p[(\boldsymbol{N} \cdot \boldsymbol{\omega_i})(1-d)+d] + K_s(\boldsymbol{\omega_o} \cdot \boldsymbol{\omega_{rd}})^{\alpha} \tag{23}$$

atenuado pelo fator do cosseno do ângulo α, o que significa que a medida que a direção de reflexão ($\boldsymbol{\omega_{rd}}$) se desvia da direção de reflexão do algorítmo propriamente dito ($\boldsymbol{\omega_o}$), o valor retornado pela BRDF decresce.

Por ser um modelo criado para um *pipeline* diferente, utiliza de artifícios para simular todas as caracterísitcas da reflectância de um material especular, que o autor subdivide em três componentes na equação do seu modelo: Uma componente *ambiente*, para representar a contribuição energética de todo ambiente em sua volta, simulando efeitos de *iluminação global* e uma constante dessa reflexão difusa e, por fim, o termo especular, que é uma porção da constante especular (K_s), com respeito ao produto interno

entre os vetores de reflexão ($\boldsymbol{\omega_r}$) e o vetor da direção de observação ($\boldsymbol{\omega_{rd}}$, gerado pela amostragem) que retorna a quantia de energia refletida na direção de observação ($\boldsymbol{\omega_{rd}}$). Para adaptar o modelo de *Phong* para o algoritmo do *Path Tracing*, é necessário descartar alguns fatores que se tornam redundantes na equação, uma vez que o próprio algorítmo já resolve parte da equação. Por exemplo: não há necessidade de simular iluminação global, ou uma componente difusa, restando apenas o componente especular da equação, aplicável ao algoritmo utilizado no nosso estudo, como na equação a seguir:

$$fr_{Phong}(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = K_s(\boldsymbol{\omega_o} \cdot \boldsymbol{\omega_{rd}})^{\alpha} \tag{24}$$

Onde K_s é a constante especular do material e α, o coeficiente que regula a rugosidade da reflectância especular nas direções ao redor da direção de reflexão direta da luz ($\boldsymbol{\omega_{rd}}$).

Se a equação permancer desta forma, a equação final que será utilizada pelo algoritmo, integrada ao aproximador monte carlo, seria expressaria, se utilizar também a amostragem uniforme, da seguinte forma:

$$Lo \approx Le(x, \boldsymbol{\omega_o}) + \frac{2}{N} \sum_{i=1}^{N} K_s(\boldsymbol{\omega_o} \cdot \boldsymbol{\omega_{rd}})^{\alpha} Li(x, \boldsymbol{\omega_i}) \cos\theta_i \ \boldsymbol{d\omega_i} \tag{25}$$

Com esta equação, comparando com outros modelos de reflectância especular, suas bordas ficam muito escurecidas, devido os produtos internos. Para resolver isso, para integração desta equação no integrador Monte Carlo e resolver este problema, é necessário omitir o $\cos\theta_i$, ou incluir este termo no denominador da função de reflectância de phong, modificando a equação 24 modificando-a na equação a seguir:

$$fr_{Phong}(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{K_s(\boldsymbol{\omega_o} \cdot \boldsymbol{\omega_{rd}})^{\alpha}}{\cos\theta_i}, \tag{26}$$

considerando que a distribuição de probabilidade $p(x)$ é igual a $\frac{1}{\pi}$, calcula-se a equação de *rendering*, através do estimador Monte Carlo, no algorítmo de *Path Tracing*, com a BRDF de Phong, conforme a equação a seguir:

$$Lo \approx Le(x, \boldsymbol{\omega_o}) + \frac{2}{N} \sum_{i=1}^{N} K_s(\boldsymbol{\omega_o} \cdot \boldsymbol{\omega_{rd}})^{\alpha} Li(x, \boldsymbol{\omega_i}) \boldsymbol{d\omega_i} \tag{27}$$

Com esta equação, resolve o problema do escurecimento das bordas, porém nos trás outro problema: a BRDF perde mais uma das suas propriedades condicionais para ser físicamente plausível, a reciprocidade, então, se alterar a ordem dos parâmetros da BRDF, das direções de incidência e de reflexão, o valor da BRDF é diferente, porque o cosseno de incidência vai mudar para cada direção de incidência, a não ser que os parâmetros sejam

descritos pela mesma direção.

Modelo de Cook-Torrance.

Um dos modelos especulares de reflectância fisicamente plausível que chega aos resultados mais realistas e um dos mais difundido na literatura atualmente é o modelo proposto por Robert L. Cook e Kenneth E. Torrance em [31], que como o modelo de *Oran-Nayar* é baseado na teoria das microfaces e é também um modelo isotrópico, que quer dizer que pode ser parametrizado da mesma forma da figura 11 da página 35. A teoria das microfaces, como explicado em [24], parte do pressuposto que existem objetos na natureza, que sua aparência é planar, no entanto, em escalas menores, sua superfície possui certas irregularidades (o que não é o caso das superfícies ópticamente planares, como a de lentes de telescópios que são planares em escala nanométrica) [23].

A superfície planar (no caso das superfícies especulares), mesmo apresentando irregularidades, interage com a luz da mesma forma que um espelho opticamente planar (o que faz com que a pequena micro-superfície só contribua com o valor total de energia refletida em uma única direção, na direção de reflexão e quando a direção de reflexão calculada pela amostragem, ω_o, for diferente a direção do vetor de reflexão da luz incidente, a reflectância se reduz à zero). Na teoria das microfaces, o ponto x, de intercecção, equivale a um diferencial de área, que possui diversas micro-áreas. A medida que a região for mais irregular, ou *rugosa*, a *distribuição da orientação* da direção do vetor normal m das microsuperfícies tem um desvio padrão maior, relacionado a orientação da geometria planar naquele ponto (vetor normal N), aumentando a probabilidade de modificação da direção de reflexão dos raios incidentes, e aumentando o desvio padrão desse desvio, gerando uma reflexão borrada/manchada da luz incidente [23]. As figuras a seguir ilustram a mudança da orientação e da direção dos raios refletidos a medida em que a irregularidade da superfície aumenta em escala nanométrica (as microssuperfícies da área diferencial do "ponto"x da integral do equação de rendering).

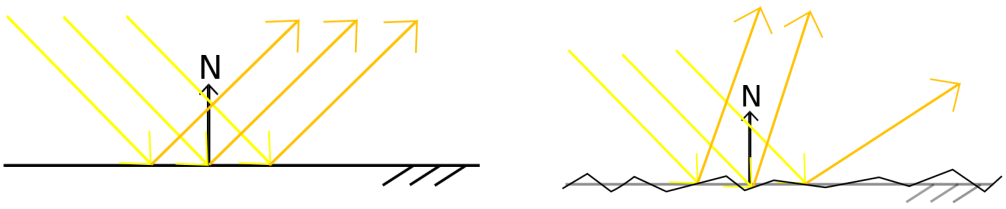


Figura 20: Superfícies opticamente planares espelhadas e superfícies planares espelhadas, refletindo a luz incidente (em amarelo) na direção de reflexão (laranja).

A forma que a luz interage com as micro-superfícies é representada no lado esquerdo

da figura 21 e a reflexão da macro-superfície com reflexão especular rugosa, devido a perturbação da direção de reflexão, consequência da irregularidade das direções dos vetores normais das micro-superfícies, é representado no lado direito da figura 21 (note que este efeito de reflexão se assemelha à reflexão do modelo de Phong).

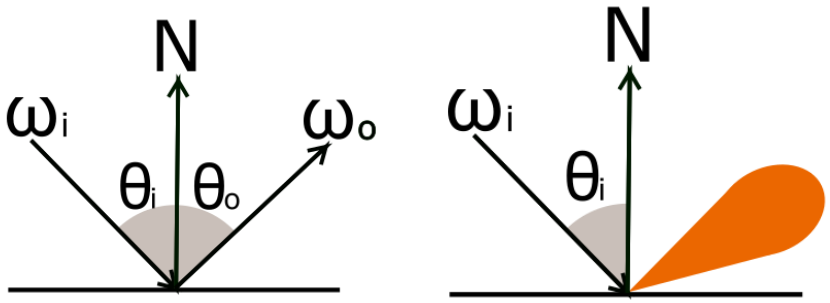


Figura 21: Reflxão em espelho opticamente planar e superfícies especulares rugosas.

Além disso, a pressuposição da presença das micro-superfícies, nos leva a inferir outros três fenômenos: os dois primeiros dizem respeito à contribuição direta da luz incidente, enquanto a ultima, diz respeito à contribuição indireta da luz. São eles: *sombreamento*, *mascaramento* e *múltiplas reflexões*, as quais foram explicados na sessão do modelo de Oren-Nayar.

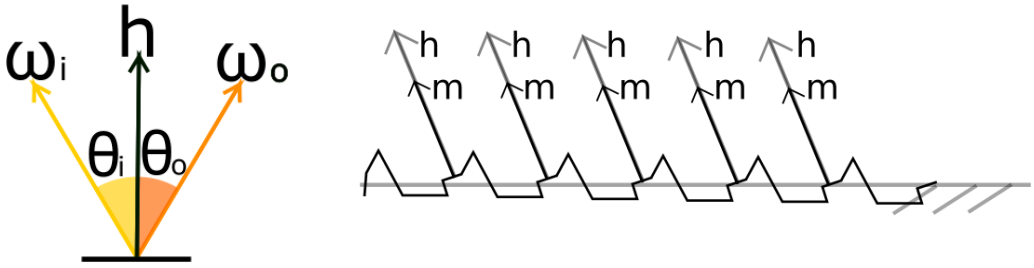


Figura 22: Vetor *halfway* e microfaces da microgeometria que formam vetores normais m, com a mesma direção e sentido do vetor h.

Para apresentar a aplicação da teoria das microfaces, na equação do modelo do *Cook-Torrance* (equação 28), é necessário apresentar uma nova variável presente na equação 28, o *halfway-vector* ou, traduzindo, vetor meio-do-caminho, representado pela letra $\boldsymbol{h}$ na mesma equação e na figura 22. Como as micro-superfícies são consideradas como espelhos planares, mesmo que pequenos, então a luz incidente na direção $\boldsymbol{\omega_i}$ vai ser refletida totalmete apenas na direção de reflexão ($\boldsymbol{\omega_o}$), que formará com o vetor normal da micro-superfície ($\boldsymbol{m}$) o mesmo ângulo (θ_i) que o vetor $\boldsymbol{m}$ formar com a direção de incidência de luz ($\boldsymbol{\omega_i}$), ou seja, para que exista contribuição de energia na direção que vemos ($\boldsymbol{\omega_o}$), é necessário que exista uma contribuição de energia incidente na direção de chegada ($\boldsymbol{\omega_i}$),

em que o vetor $\boldsymbol{h}$ é o *meio-do-caminho* e, analaogamente, só existe contribuição energética com o valor final da BRDF naquelas micro-superfícies que a direção do seu vetor normal ($\boldsymbol{m}$), coincidirem com os vetores de meio-de-caminho ($\boldsymbol{h}$), como exposto na figura 22.

Além disso, este modelo das microfaces deve, de alguma forma, retornar a parcela de energia que foi refletida e refratada, dada uma energia incidente, para o caso dos materiais que absorvem, ou transmitem a energia incidente, ou seja, retornar uma porcentagem de quanto da energia incidente é refletida, ou refratada (neste modelo, a energia refratada é totalmente absorvida pelo meio, já que estamos interessados em representar os metais como o cobre e o ouro).

O modelo de *Cook-Torrance* é descrito pela seguinte equação:

$$fr_{Cook-Torrance}(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{F(\boldsymbol{\omega_i}, \boldsymbol{h})G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{h})D(\boldsymbol{h})}{4(\boldsymbol{n} \cdot \boldsymbol{\omega_i})(\boldsymbol{n} \cdot \boldsymbol{\omega_o})} \tag{28}$$

Como o explicado em [23], o termo $D(\boldsymbol{h})$ é o termo da equação chamado de FDN (Função de distribuição de normal), que é uma função de distribuição probabilística que vai definir se, na microssuperfície atingida pela luz, a sua micro-face possua sua normal $\boldsymbol{m}$ com a mesma direção do vetor $\boldsymbol{h}$, isto é, se aquele ponto contribui ou não com a energia naquela determinada direção. A FDN, como mencionado em [18], é representada pela função de distribuição de Beckmann, como se encontra na equação a seguir:

$$D_{Beckmann}(\boldsymbol{h}) = \frac{1}{\pi\alpha^2(\boldsymbol{n} \cdot \boldsymbol{h})^4} exp\left(\frac{(\boldsymbol{n} \cdot \boldsymbol{h})^2 - 1}{\alpha^2(\boldsymbol{n} \cdot \boldsymbol{h})^2}\right) \tag{29}$$

ou pode ser expressa pela distribuição GGX da seguinte forma:

$$D_{GGX}(\boldsymbol{h}) = \frac{\alpha^2}{\pi(\boldsymbol{n} \cdot \boldsymbol{h})^2(\alpha^2 - 1) + 1} \tag{30}$$

Onde α é uma constante do modelo que representa a rugosidade dos materiais especulares. Para informações mais detalhadas nas equações 29 e 30, consultar [25].

Como exposto em [25], para garantir a corretude da FDN, é necessário conferir os seguintes requisitos:

A integral desta função, com relação à superfície do hemisfério Ω, o mesmo da figura 6, localizada na página 22, é menor ou igual a 1, ou seja: $\int_\Omega D(\boldsymbol{h}) \leq 1$, visto que, a área total da micro-superfície deve ser menor ou igual a sua área correspondente na macro-superfície. O valor da distribuição normal retorna um número real sempre positivo, entre zero e infinito, ou seja: $0 \leq D(\boldsymbol{h}) \leq \infty$ (como em [7]). A função de distribuição normal para superfícies polidas, ou suaves, em que o desvio padrão do vetor da microgeometria normal, em relação ao vetor $\boldsymbol{h}$ são pequenos, a função de distribuição se comporta como

os gráficos simulados no octave, demonstrados a seguir.

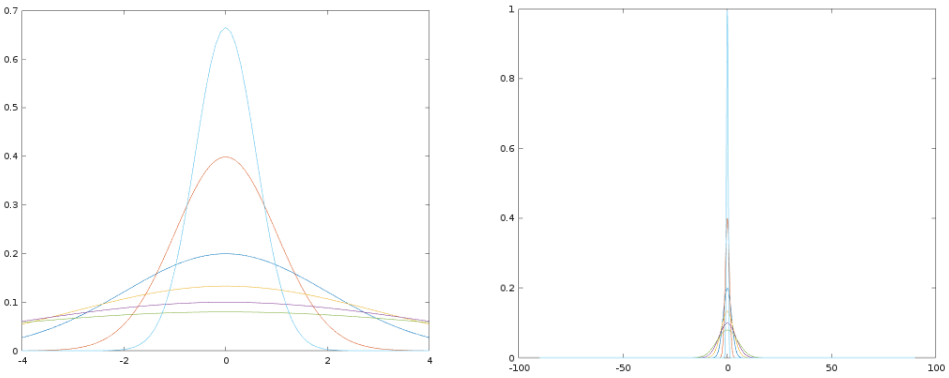


Figura 23: *Plot* da NDF.

A função $G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{h})$ é a função geométrica, que retorna o percentual (um número entre zero e um) de pontos que não estão em sombra, ou mascaradas (um número real entre zero e um), ou seja, o produto entre $D(\boldsymbol{h})$ e $G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{h})$ deve retornar os pontos ativos contribuintes com o valor final da cor do pixel, lembrando que, também como detalhado em [25], esta função respeita as propriedades de reciprocidade, ou seja, $G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{h}) = G(\boldsymbol{\omega_o}, \boldsymbol{\omega_i}, \boldsymbol{h})$. Este termo da equação da BRDF de Cook-Torrance, segundo [31], é calculado da seguinte forma:

$$G = Min\left[1, \frac{2(\boldsymbol{N}\cdot\boldsymbol{h})(\boldsymbol{N}\cdot\boldsymbol{\omega_o})}{(\boldsymbol{\omega_o}\cdot\boldsymbol{h})}, \frac{2(\boldsymbol{N}\cdot\boldsymbol{h})(\boldsymbol{N}\cdot\boldsymbol{\omega_i})}{(\boldsymbol{\omega_o}\cdot\boldsymbol{h})}\right]$$

.

Esta equação do termo G simula os efeitos de sombreamento e mascaramento, porém não simula as múltiplas reflexões.

A função F($\boldsymbol{\omega_i}$,$\boldsymbol{h}$) é a função de reflectância de Fresnel, que retorna a proporção da energia incidente que é refletida, ou refratada, em que a direção de reflexão é conhecida ($\boldsymbol{\omega_o}$), porém para calcular a direção de reflexão, caso seja desejado para simular materiais dielétricos, é necessário fazer o cálculo da direção de reflexão que recebe como parâmetro o vetor de incidência ($\boldsymbol{\omega_i}$), e o índice de refração dos dois meios em que a superfície em questão é fronteira. A funçao de reflectância é uma porcentagem, portanto retorna um valor entre zero e um e é dado pela seguinte equação:

$$F(\boldsymbol{l}, \boldsymbol{h}) = \frac{1}{2}\left({\boldsymbol{\omega_o}_{\parallel}}^2 + {\boldsymbol{\omega_o}_{\perp}}^2\right) \tag{31}$$

em que

$$\boldsymbol{\omega_o}_{\parallel} = \frac{\eta_t \cos(\theta_i) - \eta_i \cos(\theta_t)}{\eta_t \cos(\theta_i) + \eta_i \cos(\theta_t)}$$

$$\boldsymbol{\omega_{o\perp}} = \frac{\eta_i \cos(\theta_i) - \eta_t \cos(\theta_t)}{\eta_i \cos(\theta_i) + \eta_t \cos(\theta_t)}$$

Baseando-se em [16], note que qualquer vetor genérico $\boldsymbol{v}$, possui duas componentes, uma perpendiculare ($\boldsymbol{v_\perp}$) e outra paralelas ($\boldsymbol{v_\parallel}$) à superfície, em que, somadas as componentes, chegamos ao valor exato do vetor original ($\boldsymbol{v} = \boldsymbol{v_\perp} + \boldsymbol{v_\parallel}$), daí surge a notação das componentes do vetor da direção de reflexão ($\boldsymbol{\omega_o}$). Lembrando que θ_i é o ângulo formado entre o vetor de inciência da luz ($\boldsymbol{\omega_i}$), θ_t é o ângulo formado entre o vetor de refração ($\boldsymbol{\omega_t}$, direção que não nos interessa, pois não será utilizado o modelo par simular materiais que possuam transparência) e o vetor normal da macro-geometria ($\boldsymbol{h}$), que nos pontos ativos, os que contribuem com o valor final da BRDF, são iguais ao vetor normal da microface da micro-geometria ($\boldsymbol{m}$). Em [7], propõe-se uma aproximação das equações de Fresnel, chamada de aproximação de Schlick, dada por:

$$F_{Schlick}(\theta) = f_0 + (1 - f_0)(1 - (\boldsymbol{n} \cdot \boldsymbol{\omega_i}))^5 \tag{32}$$

com $f_0 = \left(\frac{\eta_1 - \eta_2}{\eta_1 + \eta_2}\right)^2$, onde η_1 e η_2 são os índices de refração dos dois meios.

Ambas são maneiras distintas e viáveis de calcular a reflectância de Fresnel. Fica a cargo do programador escolher qualquer uma das equações para calcular a reflectância Fresnel, uma vez que, como exposto em [7], quando comparados, os seus resultados são geralmente bem aproximados.

A luz se reflete na superfície das microfaces, porém a reflectânia Fresnel, ou a aproximação de Schlick retornam apenas uma informação de quantidade de energia refletida, porém não informa uma direção de reflexão. Num espelho perfeito, absolutamente toda contribuição de energia é refletida na direção de reflexão, conforme exposto na figura. A equação que retorna a direção de reflexão, em [16], é dada por:

$$\boldsymbol{\omega_o} = \boldsymbol{\omega_i} - 2(\boldsymbol{\omega_i} \cdot \boldsymbol{N})\boldsymbol{N} \tag{33}$$

O denominador da equação 28, $4(\boldsymbol{N} \cdot \boldsymbol{\omega_i})(\boldsymbol{N} \cdot \boldsymbol{\omega_o})$, é um fator de correção com relação aos valores que são transformados das microfaces que compõem a micro-geometria, sobre toda a macro-geometria. Este fator não garante a inexistência de valores negativos, para isto, para manter o resultado da BRDF sempre positiva, geralmente é utilizado custosas funções como clamp, que normaliza o valor entre zero e um (maiores que zero, ou menor igual a 1, uma vez que estritamente zero também deve ser evitado no denominador, evitando incoerências matemáticas), ou aplica-se um módulo para preservar o valor sempre positivo, e em seguida, para evitar ao máximo que o valor do produto seja zero, soma-se um valor infinitamente pequeno. Como o modelo de Cook-Torrance é muito custoso, devido o tamanho das equações utilizadas no modelo, é interessante procurar maneiras

de simplificar o cálculo, reduzindo as equações através de simplificações ou aproximações como a de Schilick [23]. Para resolver isto, é proposto uma simplificação da equação 28, utilizando a seguinte aproximação matemática encontrada em [23], como também em [32], que consiste em aproveitar a relação expressa na equação 34, chegando na aproximação da equação de cook-torrance, demonstrada na equação 36.

$$\frac{G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{h})}{(\boldsymbol{N} \cdot \boldsymbol{\omega_i})(\boldsymbol{N} \cdot \boldsymbol{\omega_o})} \approx \frac{1}{(\boldsymbol{\omega_i} \cdot \boldsymbol{h})} \tag{34}$$

$$fr_{Cook-Torrance}(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{F(\boldsymbol{\omega_i}, \boldsymbol{h})G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{h})D(\boldsymbol{h})}{4(\boldsymbol{N} \cdot \boldsymbol{\omega_i})(\boldsymbol{N} \cdot \boldsymbol{\omega_o})} \approx \frac{F(\boldsymbol{\omega_i}, \boldsymbol{h})D(\boldsymbol{h})}{4(\boldsymbol{\omega_i} \cdot \boldsymbol{h})} \tag{35}$$

A aproximação de Monte Carlo para a equação de rendering, utilizando o modelo de cook torrance, é descrita da seguinte forma:

$$Lo \approx Le(x, \boldsymbol{\omega_o}) + \frac{1}{N} \sum_{i=1}^{N} \frac{fr_{Cook-Torrance}(x, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) Li(x, \boldsymbol{\omega_i}) \cos\theta_i \ \boldsymbol{d\omega_i}}{p(x)}, \tag{36}$$

onde a $fr_{Cook-Torrance}$ está descrita na equação 28 e $p(x) = \frac{1}{2\pi}$, chegando no seguinte resultado:

$$Lo \approx Le(x, \boldsymbol{\omega_o}) + \frac{1}{N} \sum_{i=1}^{N} \frac{\frac{F(\boldsymbol{\omega_i}, \boldsymbol{h})G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{h})D(\boldsymbol{h})}{4(\boldsymbol{N} \cdot \boldsymbol{\omega_i})(\boldsymbol{N} \cdot \boldsymbol{\omega_o})} Li(x, \boldsymbol{\omega_i}) \cos\theta_i \ \boldsymbol{d\omega_i}}{\frac{1}{2\pi}}, \tag{37}$$

de onde cabe ao programador escolher que função utilizar para $F(\boldsymbol{\omega_i}, \boldsymbol{h})$, $G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}, \boldsymbol{h})$ e $D(\boldsymbol{h})$ dentre as equações que foram apresentadas, a depender da aplicação e da aplicabilidade da equação e seu reflexo no desempenho do sistema como um todo.

3.2 Amostragem e cálculo das direções de reflexão.

Todas as BRDFs recebem como parâmetros duas direções, sendo uma delas a direção de reflexão que é calculada a partir da amostragem. A amostragem uniforme calcula a direção de reflexão em um ponto x em qualquer direção possível, com igual intensidade, como no sólido de revolução demonstrado na figura a seguir:

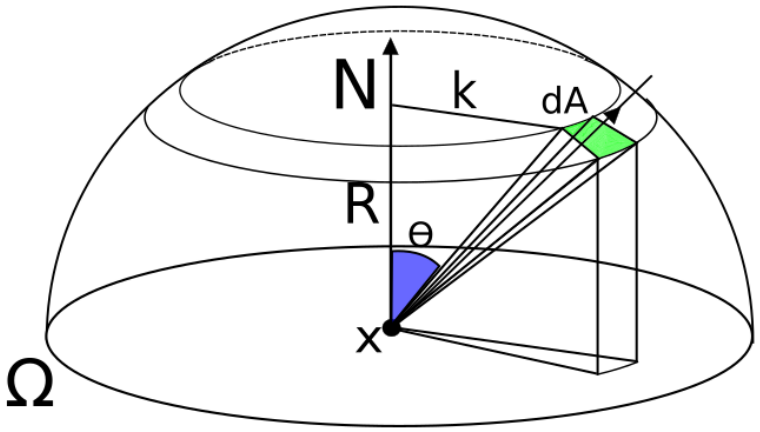


Figura 24: Superfície Ω.

Como explicado antes, para cada tipo de amostragem, existe um valor correspondente da distribuição de probabilidade do aproximador Monte Carlo. No caso da distribuição uniforme, a distribuição de probabilidade do aproximador $p(x)$ é igual a $\frac{1}{2\pi}$, devido o formato do sólido Ω que representa todas as possíveis direções de reflexão que podem ser sorteadas nesse tipo de amostragem, conforme a figura 24. Nas sessões anteriores, foram abordadas as equações do aproximador Monte Carlo com as equações de cada BRDF, assumindo que a amostragem é a mesma (uniforme) e, portanto, $p(x) = \frac{1}{2\pi}$.

A direção amostrada no sistema local da superfície, está representada pela seta preta que cruza a área projetada diferencial (em verde) no esferorradiano da figura. Como dito anteriormente, nem todas as BRDFs distribuem raios em todas as direções uniformemente, como o modelo difuso de Lambert, isso quer dizer que, dependendo da BRDF, utilizando-se da amostragem uniforme, talvez, a direção sorteada aleatoriamente faça com que a BRDF retorne o valor zero, porque a BRDF escolhida possui muito pouco ou nenhuma radiância refletida naquela direção. Existem tecnicas de amostragem por importância, que otimiza as direções calculadas na amostragem, a fim de que a direção retorne algum valor maior do que zero, melhorando a qualidade das amostras. Para obter informações mais detealhadas sobre a amostragem uniforme, ou amostragem por importância, consultar [11].

4 METODOLOGIA

Para construir todo o pipeline de síntese de imagem, utilizando o algorítmo do *Path Tracing*, e construir a versão final do renderer utilizado nos testes, foi necessário escolher, dentre as diversas opções, um método de fazer as operações aritméticas inerentes ao algorítmo de modo que consumisse o mínimo de tempo de execução na GPU possível. Esta sessão visa comprovar a validade e a aceitabilidade dos métodos utilizados, de modo a garantir a corretude da implementação do algorítmo e sua medida de performance, demonstrando a importância e a validade deste trabalho como um todo.

4.1 Programação na GPU

Como dito antes, Matthew Scarpino [36], no primeiro capítulo, traz uma discussão do quanto a GPU (Unidade de processamento gráfico) aprimorou a performance das aplicações, mas também, neste mesmo capítulo, ele cita duas linguagens de programação das unidades de processamento das placas gráficas, o CUDA e o OpenCL (O download do CUDA e do OpenCL, e suas respectivas documentações, encontram-se em [47] e [49]). A GPU por ter uma arquitetura diferente da CPU, com mais unidades de processamento, com memória local para cada unidade e oferecer, em *Hardware*, suporte para instruções vetoriais, processamento paralelo de dados, leitura e escrita rápidas de *buffers* de dados, resumindo, suporte para pipeline gráfico em sua totalidade, torna-se uma excelente opção para a programação de qualquer pipeline gráfico, por favorecer práticas de programação paralela e por suportar todas as operações algébricas básicas entre vetores e operações aritméticas entre vetores no geral. No mesmo capítulo, o autor compara as duas linguágens acima citadas, e elege o OpenCL como a melhor opção, devido a sua compatibilidade com os diversos modelos e fabricantes de placas gráficas, enquanto o cuda é compatível apenas com as placas da NVIDIA. Em concordância com os argumentos explorados em [36], o *Renderer* e o *Profiling* (medida de performance) do trabalho aqui descrito, foi implementado na linguagem C/C++ e OpenCL.

No mesmo livro, o autor explica que ao mesmo tempo que se realiza a soma de uma variável do tipo *float* com outra variável do mesmo tipo, se realiza a mesma operação entre duas variáveis do tipo *float3* (Tipo de dado do OpenCL que nada mais é do que um vetor de três *floats*), que podem ser utilizadas como as direções dos raios que percorrem a cena do algorítmo do *Path Tracing*, por exemplo (as instruções vetoriais). Nos capítulos mais adiante, o autor trás também uma explicação dos tipos de dados que regem o funcionamento da GPU através desta linguagem, como o *Host*, o programa na CPU, como C/C++, gerencia a GPU. Estas estruturas são chamadas de plataformas, dispositivos, contextos, programas, *kernels* e filas de comando. Através destas estruturas, o programador tem a capacidade de utilizar das unidades de computação e seus respactivos *work-items* para

fazer um trabalho equivalente a laços de repetição, capazes de varrer um buffer de resolução de tela, simultaneamente, num mesmo ciclo de relógio, reduzindo o tempo do processamento dos dados em comparação com as aplicação em execução na CPU, executadas proceduralmente, ou mesmo comparando com os processamentos paralelos feitos em CPU, devido o maior número de núcleos de processamento e uma arquitetura mais favorável ao paralelismo. Para infomações mais detalhadas sobre a linguagem OpenCL e seu uso para gerenciar os recursos do *hardware* da GPU, verificar [48].

4.2 Mitsuba-Renderer e Octave

Por ser um algoritmo probabilístico e envolver cálculo numérico nas diversas aproximações, torna-se difícil de avaliar a corretude do resultado final de qualquer renderizador que faça uso deste método de síntese de imagem sem compará-lo com algum resultado, utilizando os mesmos parâmetros com o objetivo de chegar aos mesmos resultados. Independente do brilho da imagem e dos modelos das cenas parecerem corretos à primeira vista a olho nú, não é prudente julgar a plausabilidade e a corretude do algorítmo apenas pela aparência do resultado final. Para avaliar a corretude da implementação do integrador Monte Carlo, do *Path Tracing* e, principalmente, dos modelos implementados no *Renderer* utilizado do nosso estudo, foi utilizado o *Mitsuba Renderer*, que é um renderizador difundido na computação gráfica e contempla uma variedade de materiais implementados e disponíveis para testes. Geradas as imagens nos dois renderizadores (o desenvolvido para gerar os testes do presente estudo e do Mitsuba), se extrai informação de alguns *pixels* no sistema de cor *RGB* (vermelho, verde e azul) e faz-se uma comparação dos gráficos obtidos com essas informações no Octave, que é amplamente utilizado na computação para avaliação de dados e comparações de sinais na computação como um todo.

4.3 *Profiling*

Existem diversas maneiras diferentes de medir performance das equações de BRDF no contexto do algorítmo escolhido. Para cada fabricante de GPU, existem profilers à disposição para medir tempo de execução das aplicações executadas na mesma, a exemplo do *Intel® Graphics Performance Analyzers*, que é compatível com a placa gráfica utilizada nos testes. Porém, como será explicado no capítulo a seguir, o OpenCL já oferece suporte a profiling e oferece uma medida de performance não só da aplicação como um todo (uma vez que o profiling é programável, dando ao programador maior controle do que está sendo calculado), envolvendo no cálculo do tempo não somente as operações dos modelos de BRDF, mas de todas as chamadas recursivas e calculos aritméticos inerentes do algorítmo em si. Como o nosso trabalho visa comparar a performance de cada função de BRDF e o suporte do OpenCL à medida de performance dos *kernels* (que são as

funções executadas nas unidades de processamento das placas gráficas, no escopo desta linguagem), a solução mais simples foi utilizar o profiler próprio da linguagem a fim de reduzir ruído da medida de performance e excluir do cálculo do tempo as outras operações que não são relacionadas a cada modelo de material isoladamente (como o tempo de lançar os raios em cada pixel), como será mais detalhado na sessão a seguir.

5 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Esta sessão está subdividida em duas partes, a primeira trata da validação dos modelos implementados e a segunda da avaliação da performance dos mesmos modelos, fazendo uma discussão de custo-benefício entre os modelos de brilho especular e os modelos de brilho difuso. Esta discussão é muito relevante, uma vez que o algorítmo do *Path Tracing* exige alto custo computacional e a escolha de um modelo a ser implementado pode tornar-se difícil e custar tempo para implementar, testar e escolher. Esta sessão visa concluir nosso trabalho com esta discussão, a fim de oferecer ao leitor um guia de qual modelo especular e qual modelo difuso deve ser implementado, conforme o que importar mais numa situação de conflito de escolha: performance ou Realismo das imagens sintetizadas.

5.1 Validação dos Modelos de BRDF

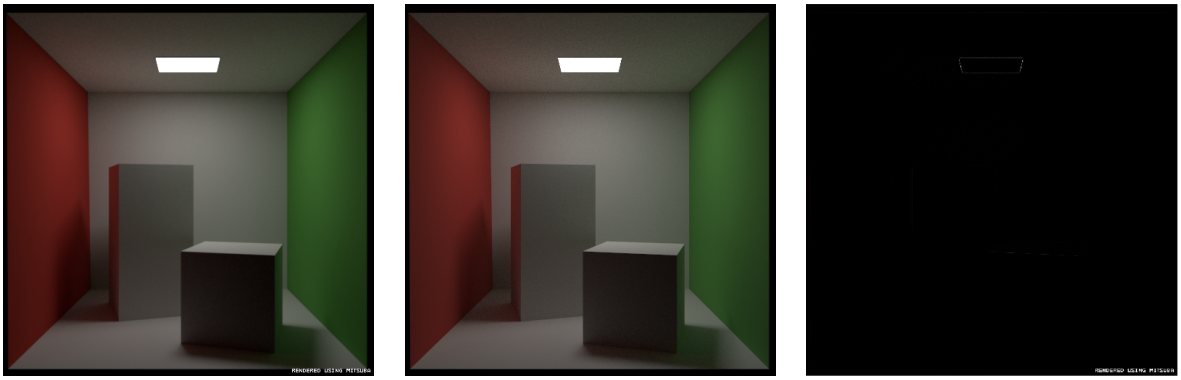
É importante relembrar que o método de síntese de imagem escolhido no nosso estudo é probabilístico, isto quer dizer que por mais que todos os testes sejam realizados na mesma cena, nas mesmas condições e mesmos materiais, podem (e muito provavelmente vão) ocorrer ruídos de maneiras diferentes a cada teste, mas também é um método numérico de resolver equações integrais que corrige seu erro a medida que o número de amostras vai aumentando, fazendo com que, com o tempo, o erro inserido aleatoriamente, vá deixando de importar para a média. Assim, o resultado final, conforme se verificará nos testes adiante, ou seja, como explicado em [43], é um método que converge para um valor "ótimo"a medida que o tempo passa. Este ideal de valor ótimo é indefinível e inatingível, pois, para atingí-lo se faz necessárias infinitas amostras. Porém, as imagens utilizadas para os testes foram recolhidas com um número considerável de amostras (o número de amostras é informado na sessão de cada teste), a fim de reduzir o erro e o ruído o máximo possível comparando-as com o resultado de outro *renderer* conhecido e aceito na computação gráfica, como dito em [35].

Para realizar a comparação pixel a pixel, é necessário configurar os dois renderizadores para que eles possam gerar exatamente os mesmos resultados, utilizando os mesmos parâmetros para que não haja distinção das imagens geradas pelos dois renderizadores. Com a especificação do integrador (*Path Tracing*) e seu número de recursões por raio, tipo de câmera, perspectiva ou ortogonal, posição da câmera, gráu de abertura da câmera perspectiva ("fov"no script xml), as constantes difusas e de emissão de luz dos objetos emissivos, com estes mesmos valores em quaisquer dois renderizadores, utilizando o mesmo cálculo de amostragem dos raios, que utilize o mesmo integrador e o mesmo tipo de câmera e a mesma BRDF implementada, devem gerar resultados praticamente idênticos.

Durante a etapa da validação, serão renderizadas em cada *renderer* duas imagens de mesmo tamanho (de resolução 512x512 pixels), nas mesmas configurações: todas com a condição de parada das chamadas recursivas do algorítmo constante igual a 5 chamadas dos raios secundários, utilizam de amostragem direta da luz, assim como no Mitsuba [45], e para conferir o a semelhança visual dos resultados gerados, serão comparadas as imagens sintetizadas pelos dois renderizadores no Octave, afim de analizar cada faixa do sistema de cor RGB de cada pixel de uma linha coletada para amostra dentre as 512 possíveis. Como são imagens de resolução 512x512, seriam necessários 512 gráficos para analisar a figura toda, ou um gráfico 3D, mas para aprimorar a visibilidade dos dados coletados, é conveniente que seja apenas uma linha, porque geram comparações mais visíveis e de melhor qualidade. Os gráficos gerados pelo matlab desta sessão, terão dois eixos: O eixo horizontal serão os pixels da linha escolhida para ser amostrada (que será especificada nas legendas dos gráficos), variando de zero até um e o eixo vertical representará a média das radiâncias que atingiram aquele pixel amostrado, sendo sempre positiva. A única coisa que será diferente entre os modelos, é o número de amostras (quantos raios serão disparados) utilizadas por pixels, que será especificado nas legendas de cada teste, e que os materiais difusos foram amostrados com amostragem uniforme, o que é mais adequado para o modelo de Lambert, o que também não é grande prejuízo para o modelo de Oren-Nayar com fator de rugosidade igual a zero, enquanto os materiais especulares, serão utilizadas amostragens por importância dos raios secundários, para que sejam necessárias menos amostras por pixel para atingir o valor esperado (o que é um problema para os modelos especulares).

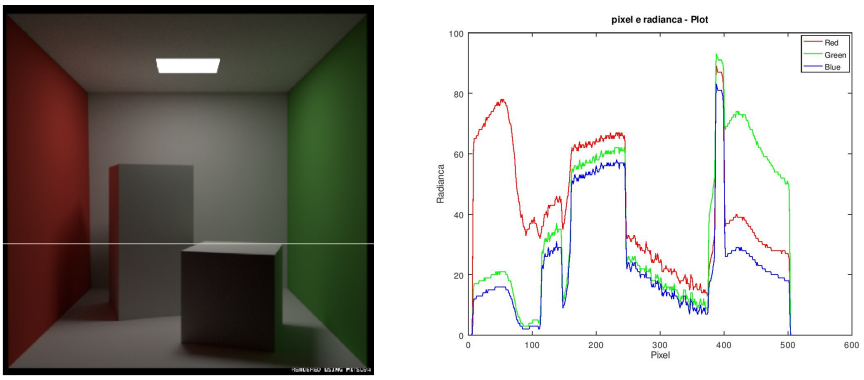
Validação da BRDF de Lambert

Utilizando a cena intitulada Cornell Box, utilizada em [9] para testar materiais difusos, foi feita uma comparação da mesma cena no Mitsuba Renderer e no Renderer desenvolvido no nosso estudo, ambas com todos os materiais difusos de Lambert e 512 amostras por pixel. Nestas configurações, foram obtidos os resultados das figuras 25 e 26.

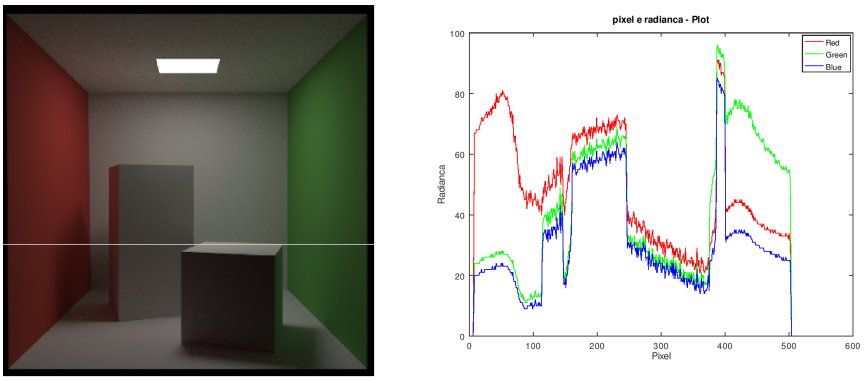


(a) (b) (c)

Figura 25: Imagem renderizada no Mitsuba (a), imagem sintetizada no renderizador da atual pesquisa (b) e a diferença absoluta entre as duas imagens (c).



(a) (b)



(c) (d)

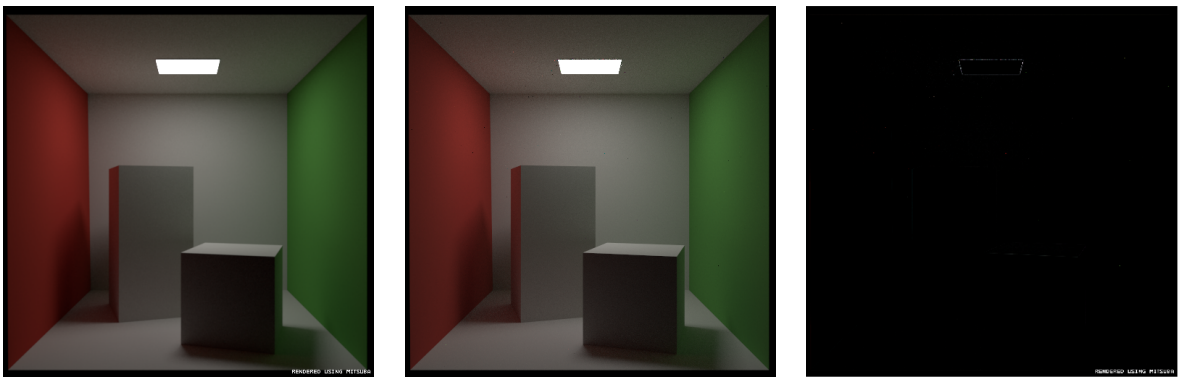
Figura 26: Imagem renderizada no Mitsuba (a) e seu respectivo gráfico gerado no Octave (b), utilizando amostras de radiância dos pixels da linha 400, destacados em (a) na cor branca. Imagem sintatizada pelo renderizador desenvolvido na presente pesquisa (c) e seu respectivo gráfico (d) amostrado na linha 400, destacado em (c) na cor branca. Nas imagens (a) e (c) foi utilizado apenas o modelo de Lambert.

Estas amostras de pixel da linha 400 foram exibidas nas três faixas de cor do sistema

RGB, separadas, cada faixa representada em sua cor. Com estes testes e a semelhança dos gráficos das saídas do mitusba *renderer* e do renderizador utilizado no nosso estudo, no Octave, podemos verificar o bom funcionamento do integrador, da câmera e do modelo difuso de Lambert do renderer desenvolvido para os testes, em comparação com o Mitsuba.

Validação da BRDF de Oren-Nayar

Do mesmo modo, a seguir, encontram-se os mesmos testes, nas mesmas condições dos testes do modelo de Lambert, porém com o modelo de Oren-Nayar. Com a finalidade de assegurar maior semelhança entre os modelos para compará-los nos dois renderizadores, foi utilizado a constante de rugosidade do modelo de Oren-Nayar igual a zero (porque em [45] é assegurado que os modelos nestas condições, geram resultados idênticos, tornando isto uma boa forma de avaliar a validade dos modelos). Apenas modificando os parâmetros do material, no xml do Mitsuba e no nosso renderer, para simular a BRDF difusa de Oren-Nayar. Com isto, foram obtidos os resultados da figura 27 e as comparações dos resultados no Octave mais adiante, na figura 28.



(a) (b) (c)

Figura 27: Imagem renderizada no Mitsuba (a), no renderizador da presente pesquisa (b) e a diferença absoluta entre as duas imagens (c).

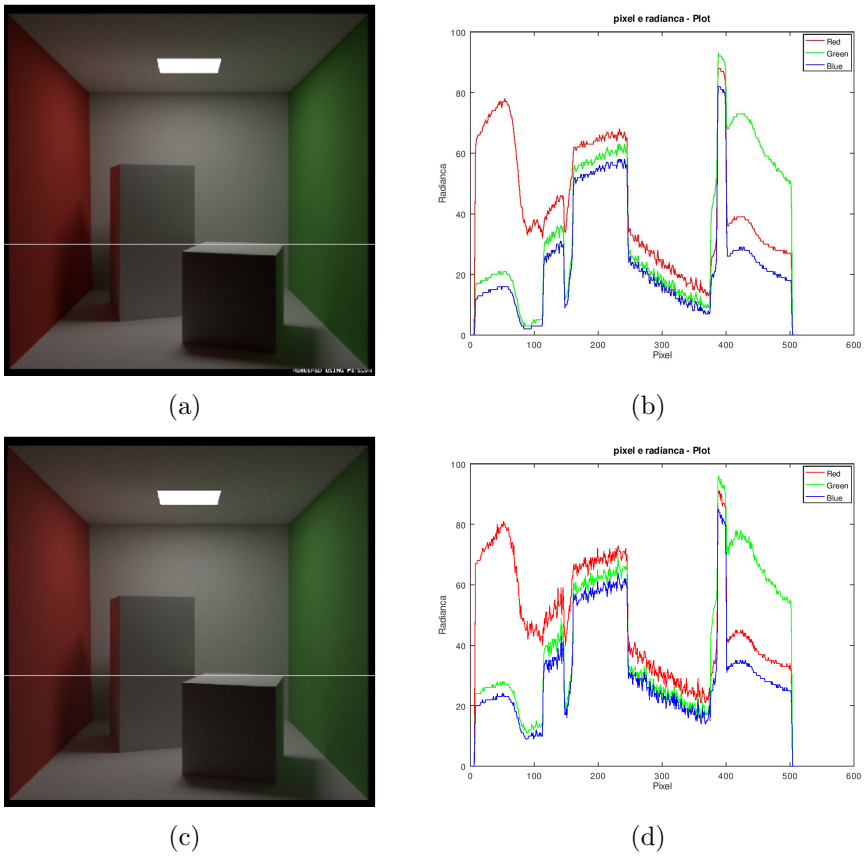


Figura 28: Imagem renderizada no Mitsuba (a) e seu respectivo gráfico gerado no Octave (b), utilizando amostras de radiância dos pixels da linha 400, destacados em (a) na cor branca. Imagem sintatizada pelo renderizador desenvolvido na presente pesquisa (c) e seu respectivo gráfico (d) amostrado na linha 400, destacado em (c) na cor branca. Nas imagens (a) e (c) foi utilizado apenas o modelo de Oren-Nayar (com as mesmas configurações de amostras por pixels, profundidade da recursão e resulução das imagens dos testes do modelo de Lambert).

Como podemos observar, os resultados para o modelo de Oren-Nayar são praticamente idênticos aos resultados obtidos no modelo de Lambert, efeito previsto para os pontos da superfície de materiais difusos, descritos pela equação aproximada do modelo, tal como descrito em 21, utilizando o termo de rugosidade igual a zero.

Para concluir a validação do modelo de Oren-Nayar, a seguir, se encontra uma comparação entre os modelos de Lambert e o modelo de Oren-Nayar, no renderer desenvolvido neste estudo e no Mitsuba, como em [26], de onde foi retirada a equação do modelo implementado nos resultados deste estudo.

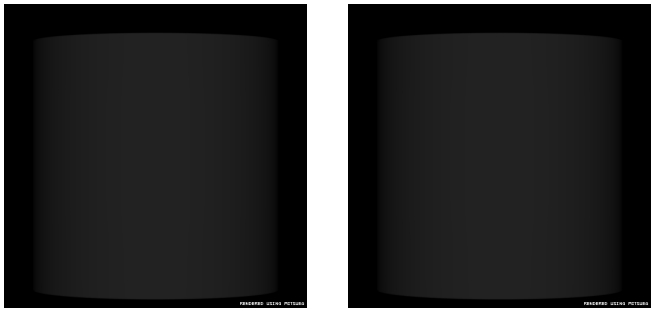


Figura 29: Comparação gráfica dos modelos de Lambert e de Oren-Nayar em cena de um cilindro no Mitsuba, com luz retângular posicionada atrás da câmera (imagens produzidas com 16 amostras por pixel).

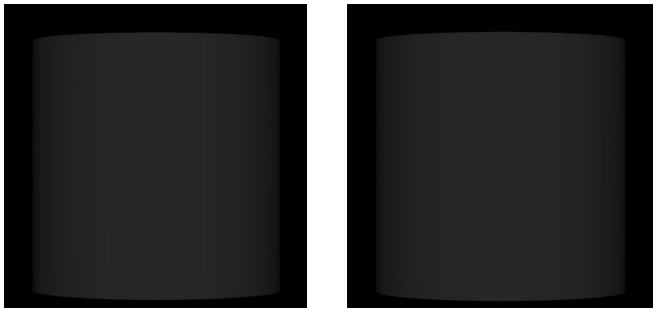


Figura 30: Comparação gráfica dos modelos de Lambert e de Oren-Nayar em cena de um cilindro no *renderer* nas mesmas condições do teste anterior.

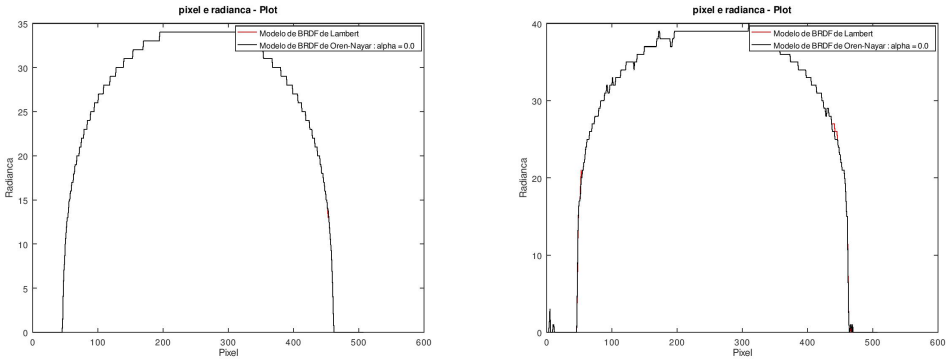


Figura 31: Comparação dos dois modelos equivalente à figura 29 e 30, no otave (amostras coletadas na linha 400).

Na figura 31, podemos observar os gráficos da linha 400 em escala de cinza do lado esquerdo, comparando as duas imgagens do Mitsuba, que encontram-se expostas na figura 29, em seguida, do lado direito da figura 31, encontra-se a análise das amostras coletadas das imagens obtidas no renderizador desenvolvido para este estudo, encontradas na figura 30. Estes gráficos Comprovam que os resultados são literalmente identicos e os

gráficos, em escala de cinza das imagens, sobrepõe um ao outro (o modelo de Lambert e o modelo de oren-nayar nos dois renderizadores). Este resultado é o esperado, uma vez que a equação da aproximação de Oren-Nayar resulta na equação de Lambert, quando é atribuído o valor zero à constante de rugosidade, como se afirma em [45] e, com isto, conclui-se a validação dos dois primeiros modelos.

À título de curiosidade, vejamos o que acontece com o modelo difuso de Oren-Nayar, com alpha igual a 40 graus, com e sem o termo especular.

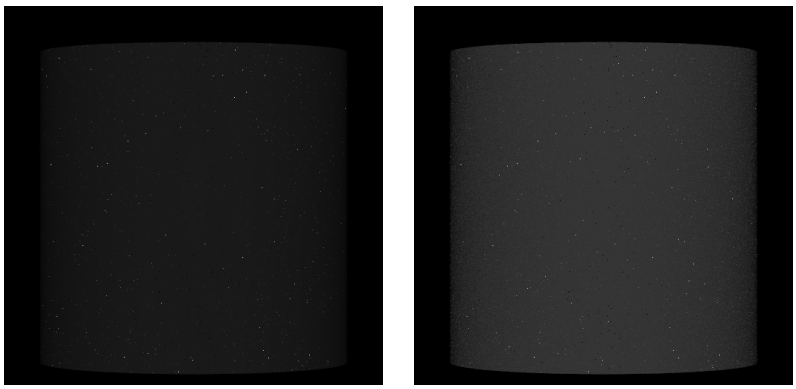


Figura 32: Comparação do mesmo modelo, nas mesmas condições das figuras 29 e 30, porém com constante de rugosidade igual a 40 graus. À esquerda, o modelo com a equação proposta neste estudo e a à direita com a mesma equação, porém, sem descartar alguns termos da aproximação utilizada em nosso estudo.

Esta variação da equação deste modelo torna-se ineficiente para as aplicações em tempo real, devido o elevado custo computacional das operações que são necessárias para simular este brilho, conforme se explica em [26], onde as duas equações que obtem os resultados das figuras 32 são apresentadas. Deste modo, a aplicação da equação dos resultados obtidos na figura do lado direito provocam uma queda de performance à aplicação, inviabilizando-a na sua integração no algorítmo nas eplicação em tempo real, apesar de gerar resultados mais claros, conforme o que se espera do modelo sem aproximação.

Validação da BRDF de Cook-Torrance

Utilizando uma cena semelhante à Cornell Box, mencionada anteriormente, porém com alguns objetos diferentes, demos continuidade com nosso estudo a fim de testar a validade da implementação dos modelos com brilho especular.

Como podemos ver na referência do Mitsuba [45], é possível estudar a especificação do material denominado *rough conductor*, que é a BRDF de Cook-Torrance, portador da propriedade *material*. Esta propriedade define o índice de refração para cada faixa de

comprimento de onda. Não há outra maneira de passar para o Mitsuba a especificação do material sem que seja por palavras, que é incompatível com nosso sistema, nos obrigando a escolher um tipo de material encontrado na tabela da especificação, nos deixando sem a possibilidade de setar os valores do índice de refração idênticos para cada faixa *RGB*. Como não restaram opções, foi necessário escolher um valor aleatório para o índice e, por tentativa e erro, analisando o gráfico da faixa *RGB* para a cena no renderizador desenvolvido nesta pesquisa e no Mitsuba, a fim de ajustar o índice de refração e encontrar curvas mais semelhantes, seguimos com nossos primeiros testes.

Como os índices são em ponto flutuante, é muito difícil chegar no valor final idêntico ao especificado no Mitsuba, porque seria necessário acertar exatamente as casas decimais dos três índices, nos deixando com uma maior margem de imprecisão. Já que o valor do índice de refração no nosso *renderer* é representado por uma tupla de três floats, foram realizados 5 testes ajustando estes três valores, até chegar ao resultado final. A seguir, encontra-se os gráficos de cada faixa (de espectro azul, vermelho e verde) de cada tupla dos índices escolhidos até atingir um valor considerado aproximado, a fim de mostrar os resultados dos ajustes e justificar a diferença dos dois sinais em alguns pontos, devido a diferença dos índices de refração, já que os valores do índice do ouro são desconhecidos. Os gráficos coloridos, são as faixas das imagens geradas pelos testes desta pesquisa e, destacado em preto, as faixas da imagem gerada no Mitsuba, tomada como parâmetro para comparação, estas seis imagens utilizada nos testes, encontram-se na figura a seguir e os gráficos encontrados com cada comparação se encontra mais adiante.

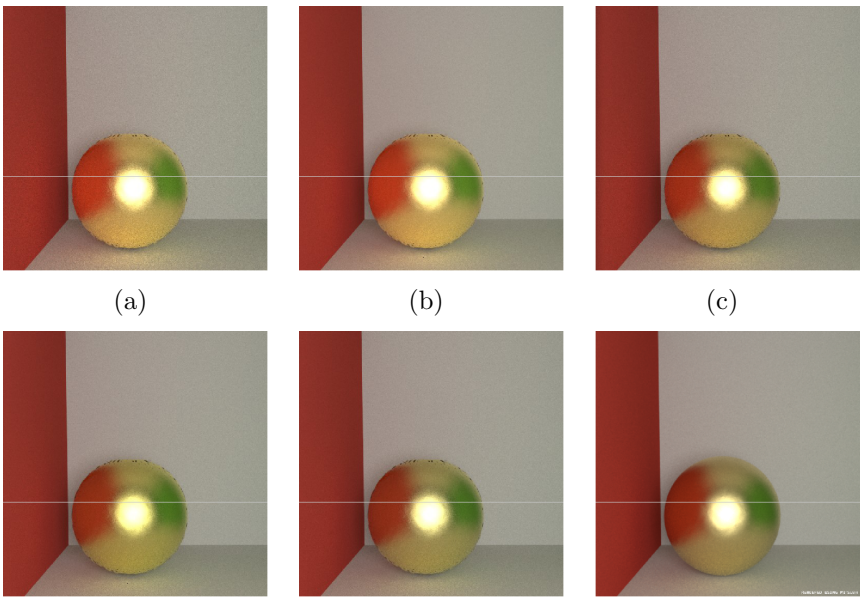


Figura 33: Imagens renderizadas no *renderer desenvolvido na presente pesquisa* e utilizadas para gerar os gráficos das figuras 34, 35 e 36

Estes gráficos das figuras 34, 35 e 36 foram esboçados da seguinte forma: A imagem 33(f) da figura 33 é a base tomada como referência para atingir o valor ótimo (gráfico em preto nas faixas vermelho, verde e azul, correspondentes aos resultados do modelo nas três faixas no Mitsuba). As imagens que a antecedem (33(a), 33(b), 33(c), 33(d) e 33(e)), geradas pelo renderizador desenvolvido na presente pesquisa, foram comparadas uma por uma, na mesma ordem que foram apresentadas, com o resultado obtido no Mitsuba (33(a)) e os resultados de cada comparação são exibidos na mesma ordem de apresentação nas três faixas de cor, do sistema de cor *RGB* (representadas nos gráficos das imagens a seguir coloridos na cor equivalente à faixa em questão: vermelho para a faixa vermelha, verde para a faixa verde e azul para a faixa azul).

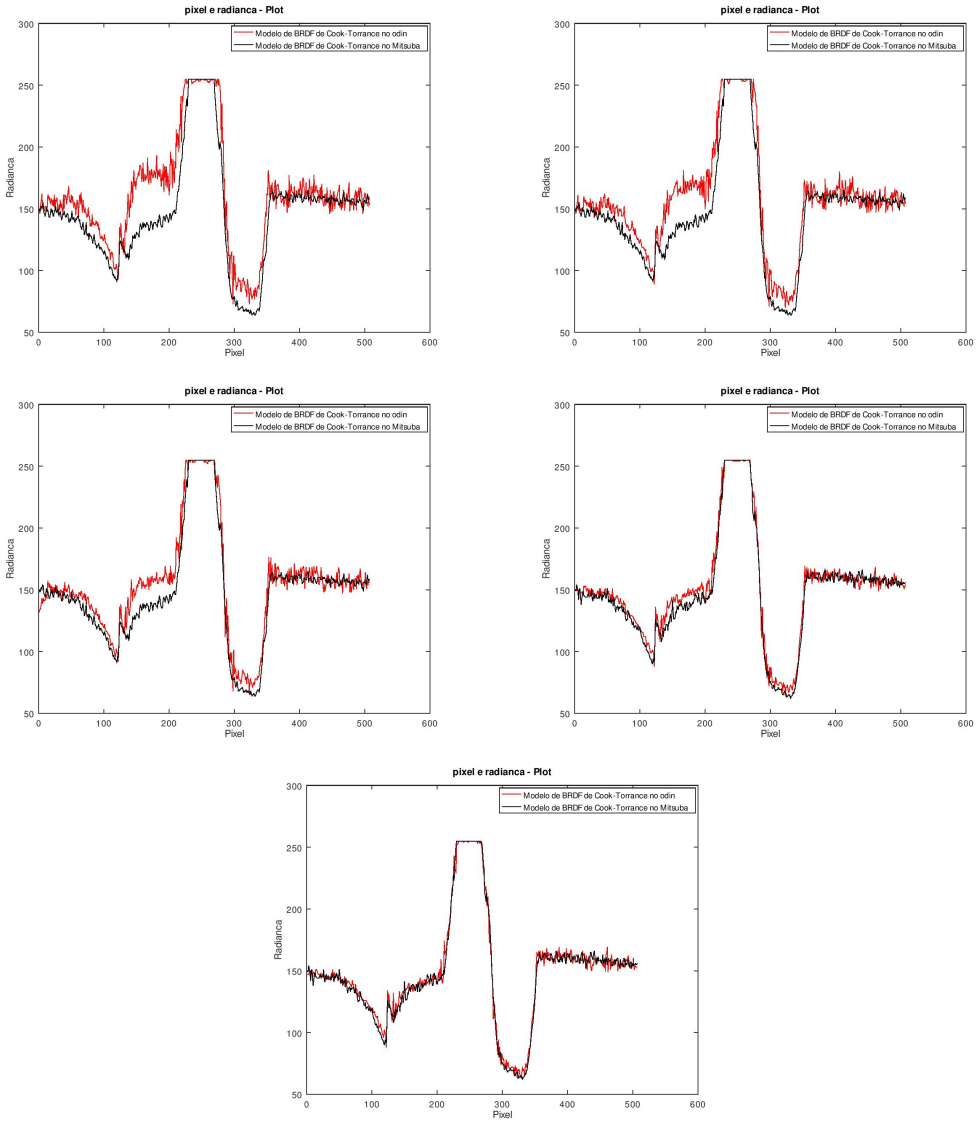


Figura 34: Comparação da faixa vermelha da cena no Mitsuba (em preto), com a faixa vermelha da cena sintetizada pelo renderer produzido no nosso estudo, ambas amostradas na linha 331, coletadas das imagens da figura 33, com 1300 amostras por pixel cada.

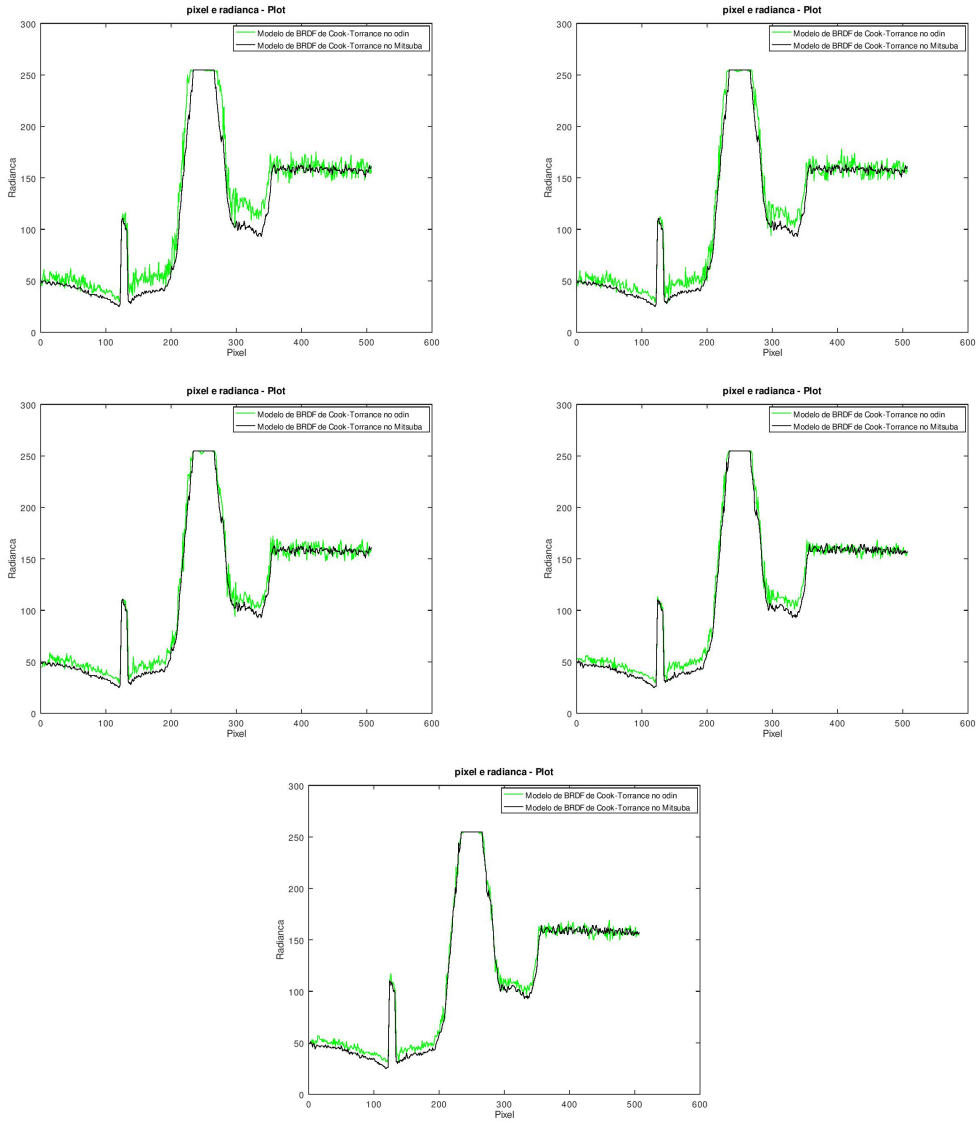


Figura 35: Comparação equivalente à figura 34, para a faixa verde.

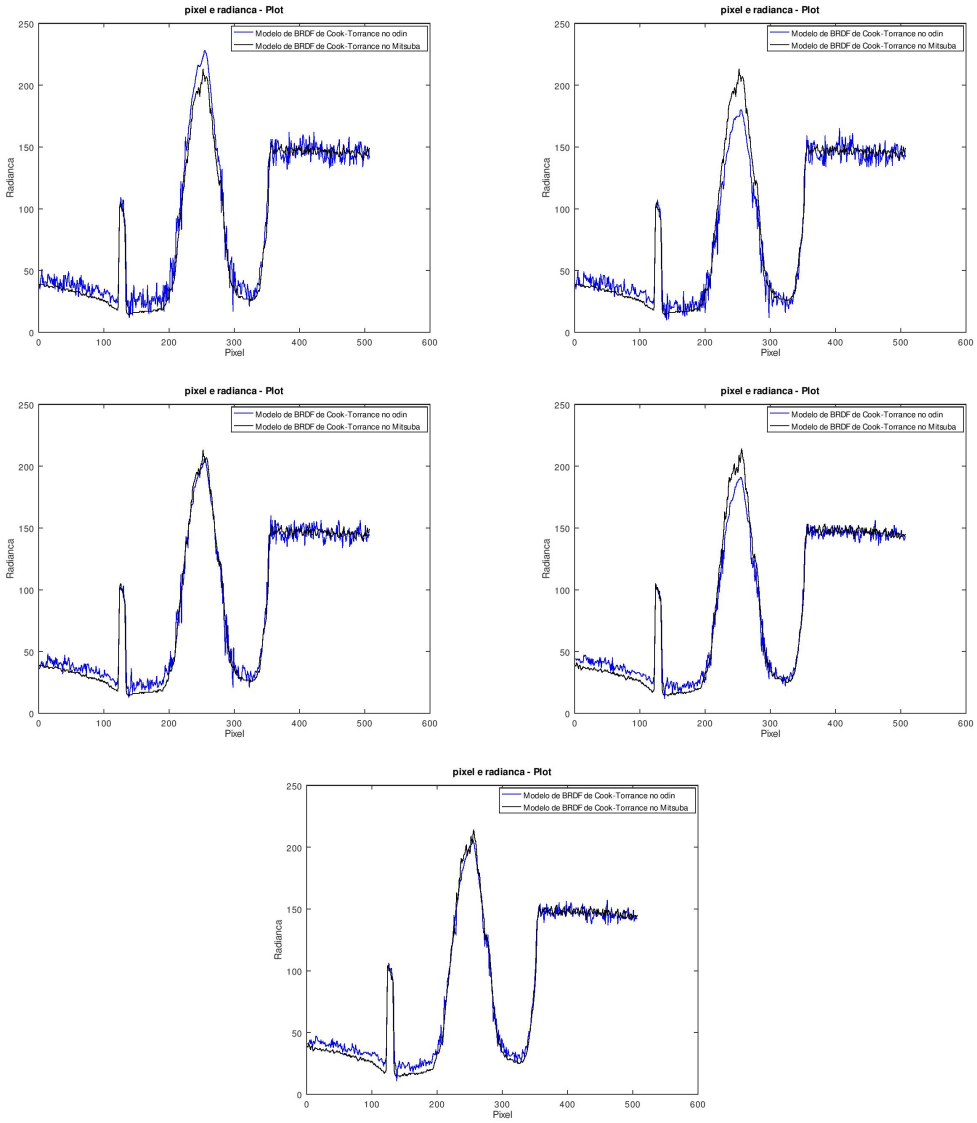


Figura 36: Comparação equivalente a figura 34, para a faixa azul.

Através deste teste, foi possível reduzir o erro, que crescia à medida que o valor da cor divergia do valor utilizado no Mitsuba, como foi verificado através da semelhança dos sinais digitais demonstrados nos últimos testes nas figuras acima.

Com o auxílio destes testes, obtemos a seguinte imagem final:

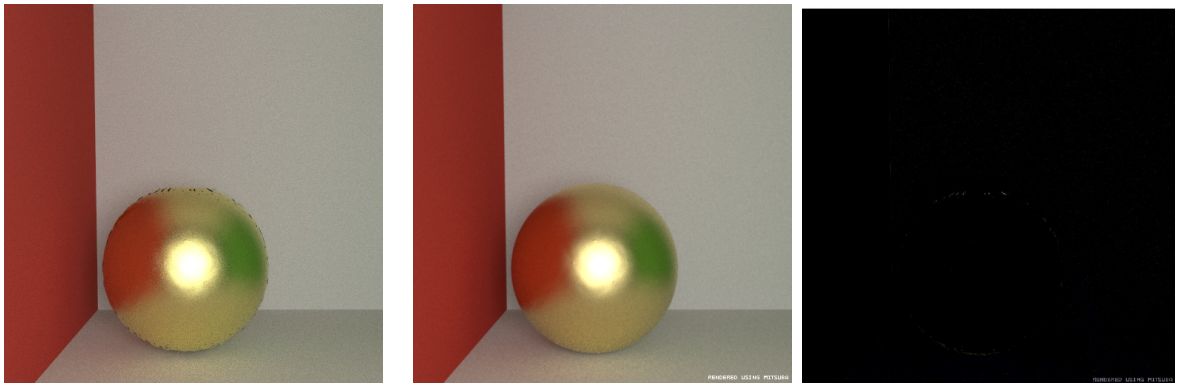


Figura 37: respectivamente, resultado obtido no *renderer* desenvolvido nesta pesquisa, no Mitsuba e a diferença entre os dois resultados.

Considerando a semelhança visual e gráfica dos modelos implementados nos dois renderers, concluímos a validade do modelo de Cook-Torrance implementado no Renderizador desenvolvido para obter os presentes resultados.

A título de curiosidade, como sabemos que existem outras maneiras de calcular a BRDF de Cook-Torrance, vejamos o que ocorre com o modelo, a ser aplicada a aproximação da equação 35 nas figuras a seguir:

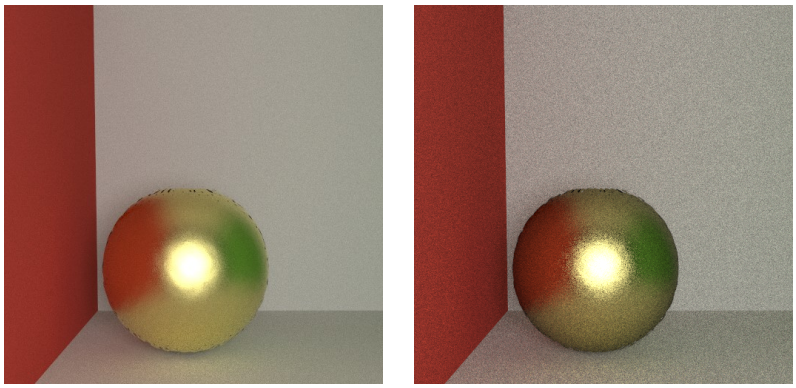


Figura 38: Resultado obtido no *renderer* desenvolvido na presente pesquisa (com e sem aproximação da equação 35).

Validação da BRDF de Phong

Como exposto em [45], o modelo de BRDF de Phong tem duas alternativas a serem utilizadas como semelhantes, para fins de comparação: o *plugin* do *rough conductor* e *roughplastic*, dependendo do tipo material. Então, com o objetivo de comparar os resultados do modelo de Phong e o modelo de Cook-Torrance, escolhemos a mesma cena do Mitsuba dos testes do modelo de Cook-Torrance, apenas alterando o parâmetro do atributo *distribution* de *beckmann* para *phong* do material da esfera. Note que ainda

utilizamos a BRDF de Cook-Torrance para avaliar a plausabilidade da BRDF de Phong em comparação com ela, apenas a função de distribuição das normais que foi alterada, na tentativa de comparar os modelos especulares. A figura 39, mostra o resultado no Mitsuba *renderer*.

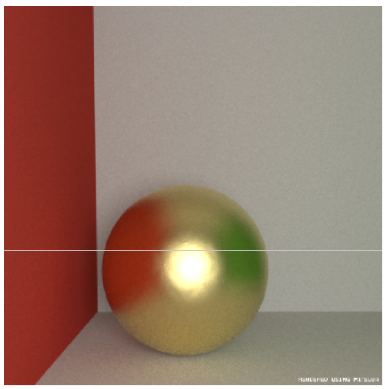


Figura 39: Resultado obtido no Mitsuba do modelo de Phong (com 1300 amostras por pixel e amostrado na linha 331, destacada em branco na figura, como no teste do modelo de Cook-Torrance).

Antes de obter o resultado final, utilizando amostragem por importância e o fator de correção da equação 27, obtivemos alguns resultados, este processo será demonstrado na figura 41.

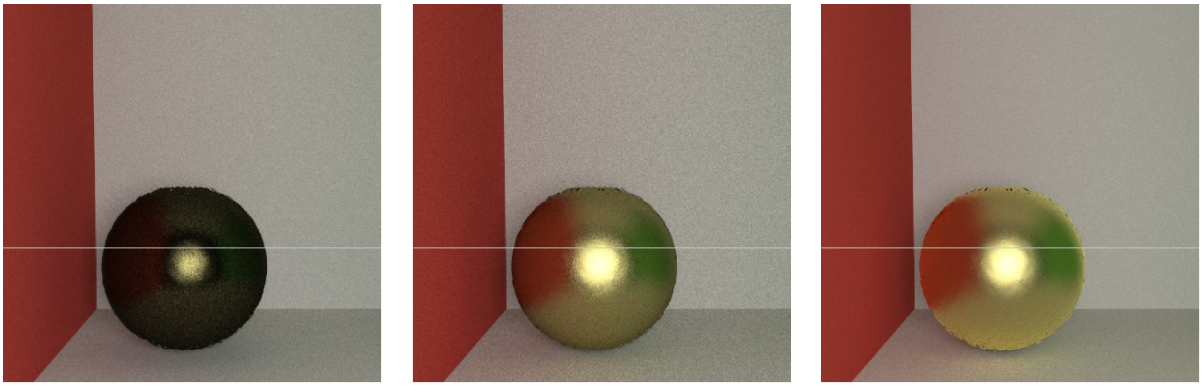
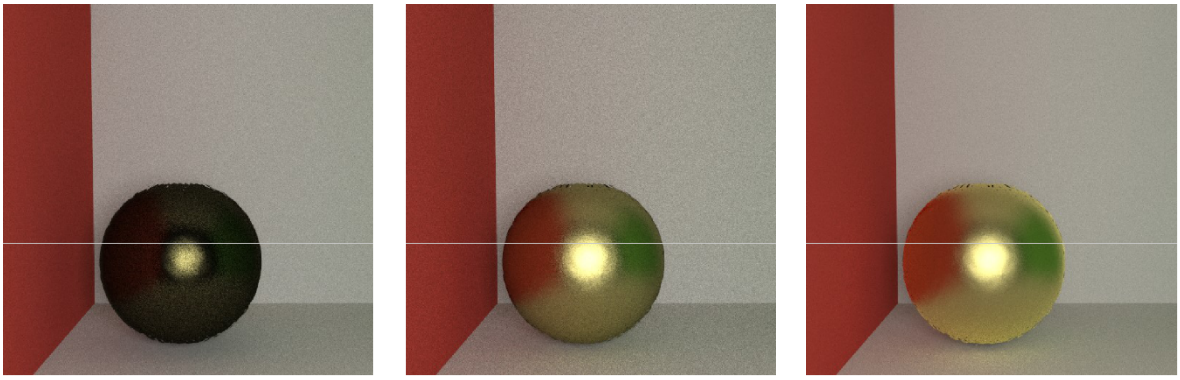


Figura 40: Resultados obtidos no *renderer* desenvolvido para esta pesquisa, do modelo de Phong (as duas primeiras imagens foram geradas com 1200 amostras por pixel e a terceira com 1300 e todas elas foram comparadas no Octave com as amostras coletadas pela linha 331, destacadas em branco nas imagens).



(a) (b) (c)

Figura 41: Resultados obtidos no *renderer* desenvolvido para esta pesquisa, do modelo de Phong (as imagens (a) e (b) foram geradas com 1200 amostras por pixel e (c) com 1300 e todas elas foram comparadas no Octave com as amostras coletadas pela linha 331, destacadas em branco nas imagens).

A figura 41(a) foi gerada utilizando a equação 25, a figura 41(b) fez uso da equação 24 e a figura 41(c), utilizando a equação 27. De modo semelhante à validação do modelo de Cook-Torrance, foi utilizanda a imagem gerada no Mitsuba como referência (figura 39, representada pelos gráficos gráficos em preto, nas figuras à seguir, para comparar com as imagens 41(a), 41(b) e 41(c), nesta mesma ordem, representadas pelas cores de cada faixa, utilizando a linha 331, destacada nas imagens em branco para coletar as amostras de cada imagem), a seguir, encontram-se os resultados obtidos em cada comparação, nas três faixas de cor *RGB*.

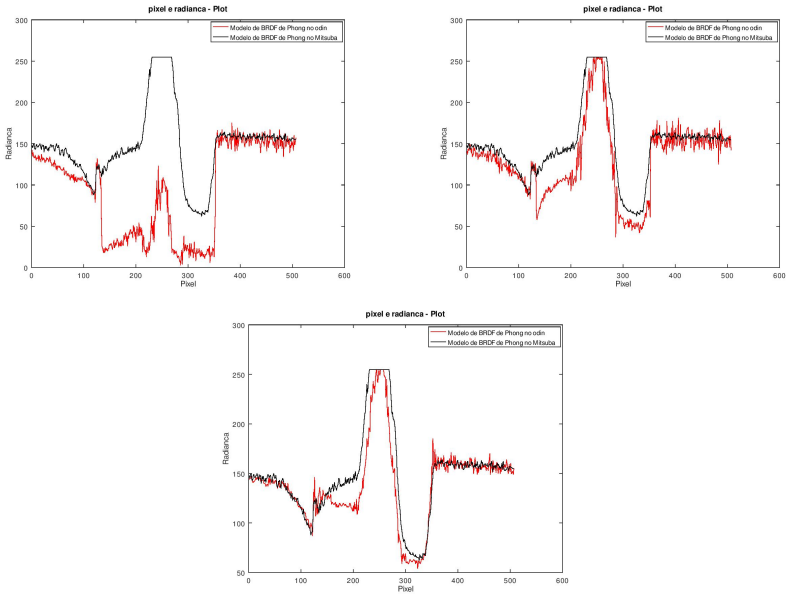


Figura 42: Comparação da faixa vermelha da cena no Mitsuba, utilizando o modelo de Cook-Torrance e a distribuição de Phong (em preto, nos gráficos), com a faixa vermelha da cena no *renderer* desenvolvido para a presente pesquisa (em vermelho, nos gráficos), utilizando o modelo de Phong, ambas amostradas na linha 331.

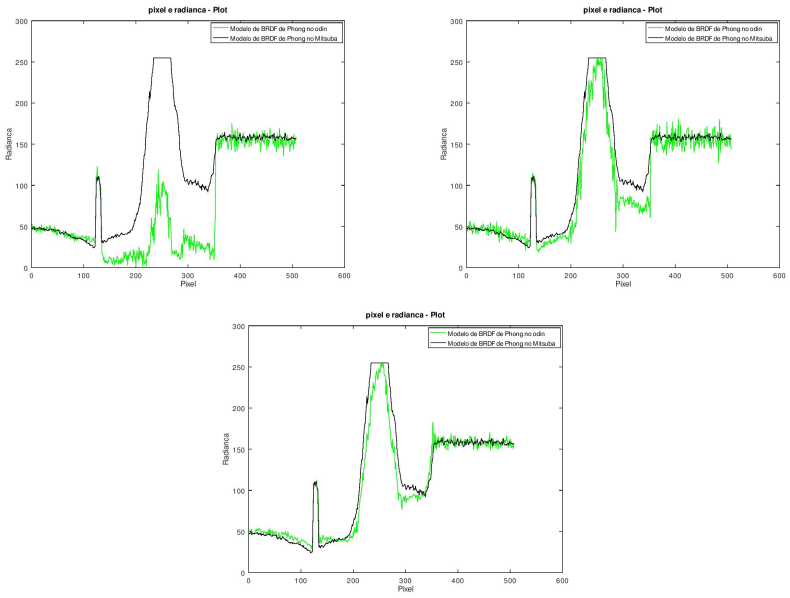


Figura 43: Comparação da faixa verde da cena no Mitsuba, utilizando o modelo de Cook-Torrance e a distribuição de Phong (em preto, nos gráficos), com a faixa verde da cena no *renderer* desenvolvido para a presente pesquisa (em verde, nos gráficos), utilizando o modelo de Phong, ambas amostradas na linha 331.

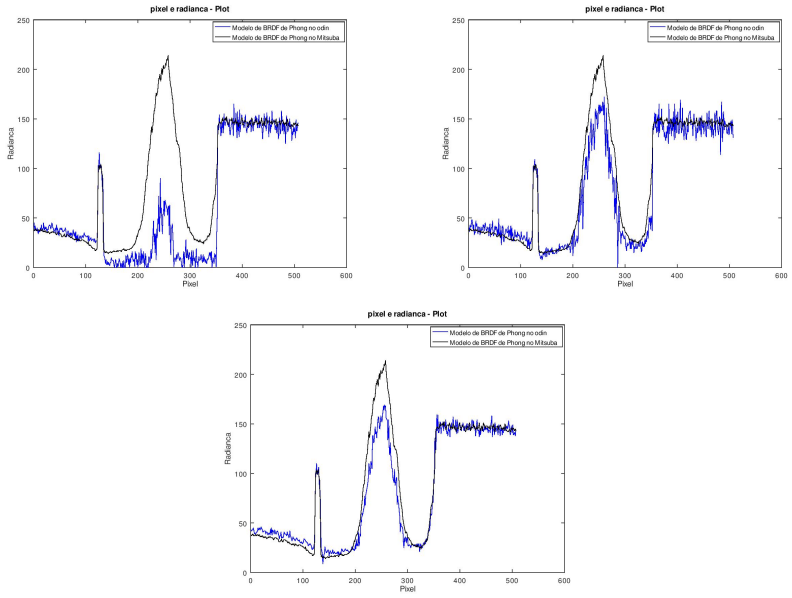


Figura 44: Comparação da faixa azul da cena no Mitsuba, utilizando o modelo de Cook-Torrance e a distribuição de Phong (em preto, nos gráficos), com a faixa azul da cena no no *renderer* desenvolvido para a presente pesquisa (em azul, nos gráficos), utilizando o modelo de Phong, ambas amostradas na linha 331.

Nas figuras 42, 43 e 44, encontram-se os gráficos do Octave para as faixas RGB das imagens da figura 41 (gráficos coloridos de acordo com a faixa de cor em comparação), com a imagem da figura 39. A imagem a seguir demonstra visualmente, pixel a pixel, a subtração das imagens da figura 41 da figura tomada como base para a comparação, gerada no Mitsuba (imagem da figura 39), a fim de reafirmar a importância da amostragem por importância explicada na sessão de validação do modelo de Oren-Nayar e sua contribuição para redução do erro, mas também demonstrar a importância do fator de correção do modelo de Phong, como demonstrado na sessão de apresentação do mesmo modelo neste trabalho.

Apesar do erro ser maior, este era o valor esperado para a BRDF de Phong, em comparação à BRDF de Cook-Torrance, principalemnte por serem modelos diferentes, mas por simularem materiais de brilho especular, deveriam ter valores bem aproximados, contudo, a BRDF de Phong não se adequa com perfeição ao comportamento da BRDF de Cook-Torrance, pelas razões que se explicarão imediatamente. Apesar de estar integrado num algorítmo de síntese de imagem baseado em física, foi criado para um pipeline de rasterização, além de que o fator de correção faz com que a BRDF não seja mais recíproca, deixando de ser bidirecional e deixa de ser uma BRDF físicamente plausível e um modelo empírico, isto quer dizer que, ela não retorna a mesma quantia de energia que um material especular físicamente plausível deveria, porque não respeita as condições das leis da física

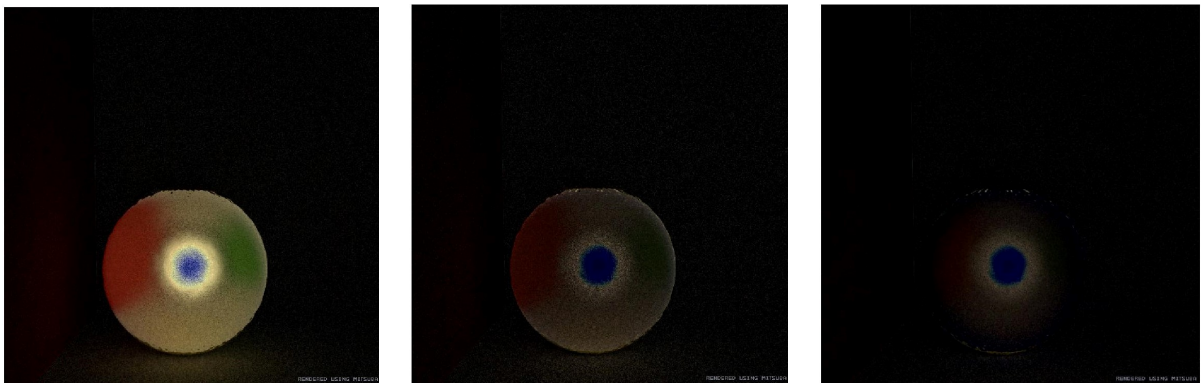


Figura 45: Imagens resultantes da subtração das imagens da figura 41 da figura 39.

com relação a conservação de energia, porém retorna resultados bem aproximados ao modelo de Cook-Torrance. Ou seja, do ponto de vista estético, visual, a BRDF de Phong chega a resultados interessantes e ligeiramente diferente dos resultados do Cook-Torrance, mas do ponto de vista físico, científico, analítico, matemático, o modelo deixa a desejar, por não ser um modelo fisicamente plausível. Então, dependendo do objetivo procurado com a implementação do modelo de Phong, poderá ser, ou não, uma boa escolha, em comparação com o modelo de Cook-Torrance e, nos resta, ainda, considerar a performance dos dois modelos para completar esta análise, como será feito na próxima sessão. Quanto à aproximação da equação 35 do modelo de Cook-Torrance, foi desconsiderada, visto que ela alcança resultados visualmente menos realistas do que o modelo de Phong, que é um modelo empírico.

5.2 Medida de Performance

A placa gráfica utilizada para os testes é a *Intel(R) HD Graphics*. Como foi explicado anteriormente, as funções da biblioteca do OpenCL foram utilizadas para medida de performance dos *kernels*, a fim de comparar o desempenho de cada equação dos modelo de BRDF.

Como a finalidade dos testes é, exclusivamente, medir o tempo das chamadas das funções das BRDFs, é necessário excluir da medida de tempo da execução dos kernels tudo que há em comum na sua execução no contexto do algorítmo: cálculo de testes de interceção com os objetos da cena, a leitura e escrita de dados em *buffers*, exibição da imagem na tela, contrução das estruturas de aceleração, etc. Para isto, é necessário remover todo o Renderer, e executar apenas os kernels das BRDFs recebendo como parâmetro apenas as informações necessárias para calcular a função de distribuição de reflectância bidirecional correspondente a cada modelo. Adiante, encontra-se um esboço do código do *Kernel* do modelo difuso de Lambert, a fim de discutir a maneira mais eficiente de medir

performance na GPU, no presente estudo.

```
__kernel void lambert_brdf(__global float *_wi,
                           __global float *_wo,
                           __global float *_fr/*BRDF*/
                           __global float3 *n/*normal da superficie*/
                           __global material *mtl/*constantes dos materiais
    */) {

     /* Converte valores do tipo de dado float, para o tipo de dado basico
    do OpenCL(float3)
     .
     .
     .
     */

     /*calcula indices do buffer object*/
   int j = get_global_id(0);
   int i = get_global_id(1);

   //CALCULO DA BRDF DE LAMBERT
   /*kd -> constante difusa do material*/
   float3 fr = (mtl->kd)*(1/M_PI_F);/*rho/pi*/

   //ATRIBUI O RESULTADO AO BUFFER OBJECT
   int offset = i*3+j*3*2D_BUFFER_RESOLUTION_WIDTH;
   _fr[offset] = fr.x;
   _fr[offset+1] = fr.y;
   _fr[offset+2] = fr.z;
}
```

Para comparar a performance do cálculo das BRDFs, o ideal seria preservar apenas o trecho do cálculo de cada BRDF, porém o compilador C e do OpenCL contém otimizadores (para melhores informações, verificar [10]) que, caso uma variavel não seja utilizada no desenvolver da aplicação, todo o calculo é desconsiderado e o *Kernel* não chega a ser executado. Então, para que o Kernel fosse executado e nenhuma otimização descartasse nenhuma operação, era necessário que a BRDF recebesse os valores como parâmetro, processasse os dados e escrevesse o resultado num *buffer object* que seria utilizado posteriormente no *Host*, por isso, o código do Kernel das BRDFs dos materiais implementados devem seguir a mesma lógica, para que o *Kernel* seja executado e a medida de performance seja realmente do cálculo integral das BRDFs.

Para melhorar a performance do profiler, uma alteração deste código deve ser feita. As constantes dos materiais não precisam ser passadas por parâmetro, podem fazer parte das equações, pois a finalidade deste teste não é calcular o brilho de um objeto e renderizar uma imagem, mas de apenas medir o tempo do cálculo. Por isso, os valores das

componentes das direções, da normal, o seu resultado e o valor das constantes é irrelevante, por isso, mantemos, apenas no código do profiler, as constantes como números fixos. Como se segue na alteração do trecho de código exibido acima, no trecho de código abaixo:

```
__kernel void lambert_brdf(__global float *_wi,
                           __global float *_wo,
                           __global float *_fr/*BRDF*/
                           __global float3 *n/*normal da superficie*/) {

    /* Converte valores do tipo de dado float, para o tipo de dado basico
    do OpenCL(float3)
    .
    .
    .
    */

    /*calcula indices do buffer object*/
  int j = get_global_id(0);
  int i = get_global_id(1);

  //CALCULO DA BRDF DE LAMBERT

  float3 fr = (float3)(0.7f,0.7f,0.7f)*(1/M_PI_F);/*rho/pi*/

  //ATRIBUI O RESULTADO AO BUFFER OBJECT
  int offset = i*3+j*3*2D_BUFFER_RESOLUTION_WIDTH;
  _fr[offset] = fr.x;
  _fr[offset+1] = fr.y;
  _fr[offset+2] = fr.z;
}
```

Repetindo o mesmo *Kernel* para os três outros modelos, modificando apenas a reflectância (f_r). Para o modelo de Lambert, f_r é descrita pela equação 16, para o modelo de Oren-Nayar é descrita pela equação 21, para o modelo de Phong, a equação 26 e, para o modelo de Cook-Torrance, é descrita pela equação 28 (todas devidamente validadas na sessão anterior). A equação da reflectância do modelo de Cook-Torrance, como visto na sessão onde foi explicado como esta BRDF é representada no contexto do *Path Tracing*, pode ser implementada de diversas maneiras diferentes, porque os termos da função de reflectância podem ser calculados de maneiras diferentes. No renderer e no no profiler da presente pesquisa foi implementada a equação de Cook-Torrance com a aproximação de Schilick (termo F) e a distribuição de vetores normais das microssuperícies (termo D) de Beckmann.

Estes *Kernels* foram executados na placa gráfica Intel(R) HD Graphics, em que o número máximo de *work-items* por *work-group* é 256. Esta placa possui 12 *compute units*,

que executa um *work-group* por vez. Isto quer dizer que é possível serem executados, no máximo, 12x256 pixels num só ciclo de clock. Para maiores informações de como obter estas informações da placa de vídeo através do OpenCL, consultar [47] e [48].

Para utilizar o máximo de unidades de processamento paralelo da GPU, para executar simultaneamente os *Kernels* e medir seu tempo, no *Host* foram alocados 12x256x4 *bytes* para o *buffer object* em que os dados de saída seriam escritos. Dessa forma, um *Kernel* não interrompe a escrita na mesma região de memória de um outro, fazendo com que sejam executados simultâneamente e ininterruptamente, garantindo atomicidade do *buffer object* na memória na GPU. Para maiores informações em como funciona particionamento de dados, gerenciando os *work-items*, verificar [19].

Disparando os 12 *work-groups*, com 256 *work-items* cada (um total de 3072 *work-items* operando em paralelo, cada um processando uma cópia do Kernel), na placa Intel(R) HD Graphics, obtivemos as medidas de performance conforme a tabela 1.

Tabela 1: *Profiling* (em milissegundos) dos modelos de BRDF no *renderer*. Respectivamente, Lambert, Oren-Nayar, Phong e Cook-Torrance Com 12 *Work-groups* de 256 *Work-items*).

Tempo	Lambert	Oren-Nayar	Phong	Cook-Torrance
Tempo mínimo	11.354 ms	37.135 ms	31.667 ms	33.437 ms
Tempo médio	12.275 ms	39.419 ms	34.239 ms	36.582 ms
Tempo máximo	50.052 ms	90.833 ms	103.384 ms	112.187 ms

A medida de tempo na GPU é contada a partir de um temporizador que, no nosso caso, decorrem 52 nanossegundos para alterar o valor do temporizador. Além dessa margem de erro de 52 nanossegundos, a medida de tempo pode ser ruidosa, por algum problema no *Hardware*. Para evitar este tipo de problema, nosso profiler dispara os 12 *work-groups* num laço de 2000 repetições, exibe o tempo máximo, o tempo mínimo e calcula a média dessas medidas de tempo a fim de aumentar a precisão dessa medida para os *kernels* de cada tipo de material.

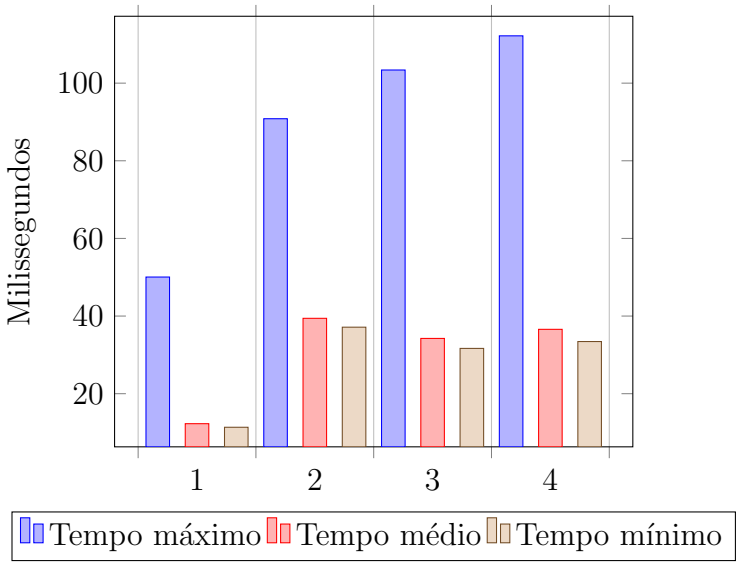


Figura 46: Histograma do Profiling dos 4 modelos implementados (1-Lambert 2-Oren-Nayar 3-Phong 4-Cook-Torrance).

Com estes dados, podemos observar que, em média, o modelo de Phong é apenas 2.343 *ms* mais rápido do que o modelo de Cook-Torrance e o modelo de Lambert é 27.144 *ms* mais rápido do que o modelo de Oren-Nayar. Como o desvio padrão do tempo máximo ficou muito alto para os quatro modelos neste teste, foram feitos dois ultimos testes (o primeiro, utilizando metade das unidades de processamento de processamento paralelo da placa gráfica disponíveis, com 6 *work-groups*, composto por 256 *work-items* cada (um total de 1536 *work-items* operando em paralelo, cada um processando uma cópia do Kernel), e o segundo, utilizando o mínimo de unidades de processamento paralelo da placa gráfica, com apenas um *work-group* com 256 *work-items*), a fim de verificar se o tempo médio é pelo menos aproximado, verificando a precisão dos resultados, com base em outros testes.

Como explicado antes, a seguir, encontram-se os resultados, primeiro, do teste com 6 *work-groups*, composto por 256 *work-items* cada, conforme a tabela e o gráfico a seguir.

Nestes testes, em média, o modelo de Oren-Nayar foi 22.336 *ms* mais lento do que o modelo de Lambert e o modelo de Cook-Torrance 0.903 *ms* mais lento do que o modelo de Phong. Para facilitar a visualização dos dados obtidos com estes testes, verificar a figura 47.

Tabela 2: *Profiling* (em milissegundos) dos modelos de BRDF no *renderer*. Respectivamente, Lambert, Oren-Nayar, Phong e Cook-Torrance Com 6 *Work-groups* de 256 *Work-items*).

Tempo	Lambert	Oren-Nayar	Phong	Cook-Torrance
Tempo mínimo	8.594 ms	29.426 ms	26.666 ms	27.240 ms
Tempo médio	10.900 ms	33.239 ms	28.855 ms	29.758 ms
Tempo máximo	39.426 ms	71.093 ms	64.583 ms	69.063 ms

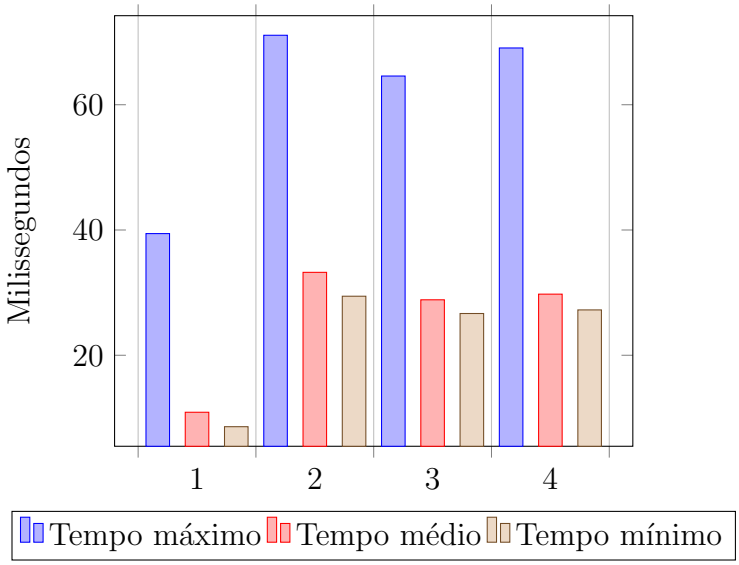


Figura 47: Histograma do Profiling dos 4 modelos implementados (1-Lambert 2-Oren-Nayar 3-Phong 4-Cook-Torrance).

Seguindo com os mesmos testes, com o mínimo de *work-items* e *work-groups* suportada pela placa utilizada nos testes (1 *work-group*, composto por um *work-item*, foram encontrados os resultados das imagens da figura 3.

Nestes testes, em média, o modelo de Oren-Nayar foi 21.199 *ms* mais lento do que o modelo de Lambert e o modelo de Cook-Torrance 1.276 *ms* mais lento do que o modelo de Phong. Para facilitar a visualização dos dados obtidos com estes testes, verificar na figura 48.

Tabela 3: *Profiling* (em milissegundos) dos modelos de BRDF no *renderer*. Respectivamente, Lambert, Oren-Nayar, Phong e Cook-Torrance Com 1 *Work-group* de 256 *Work-items*).

Tempo	Lambert	Oren-Nayar	Phong	Cook-Torrance
Tempo mínimo	5.312 ms	24.791 ms	22.968 ms	22.604 ms
Tempo médio	5.950 ms	27.149 ms	24.648 ms	25.924 ms
Tempo máximo	37.187 ms	68.593 ms	82.864 ms	57.135 ms

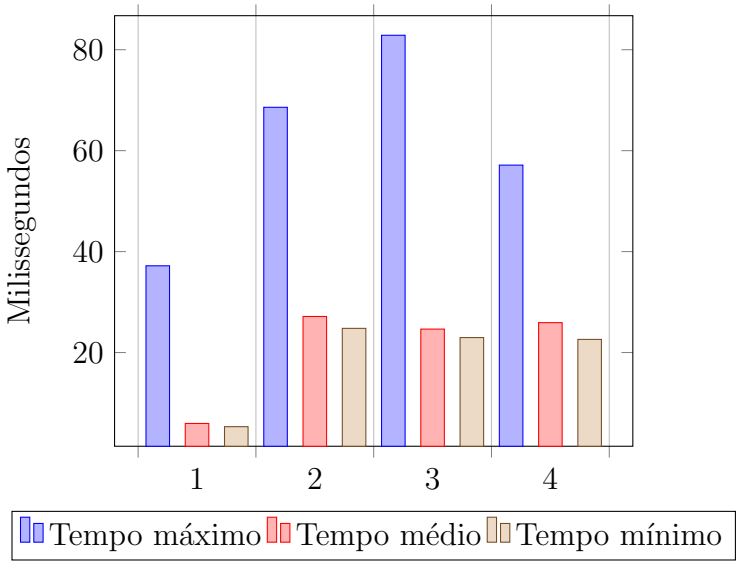


Figura 48: Histograma do Profiling dos 4 modelos implementados (1-Lambert 2-Oren-Nayar 3-Phong 4-Cook-Torrance).

Como observa-se que o desvio padrão permanece sendo elevado para os valores de tempo máximo nos três testes, resta-nos apenas, fazer uma pequena análise desses gráficos, considerando unicamente os tempos médios dos três testes (Lembrando que o primeiro teste foi feito com 12 *work-groups*, o segundo com 6 e o terceiro teste com 1, todos os três testes foram feitos com *work-groups* compostos por 256 *work-items*). Para isto, observemos o gráfico da figura 49.

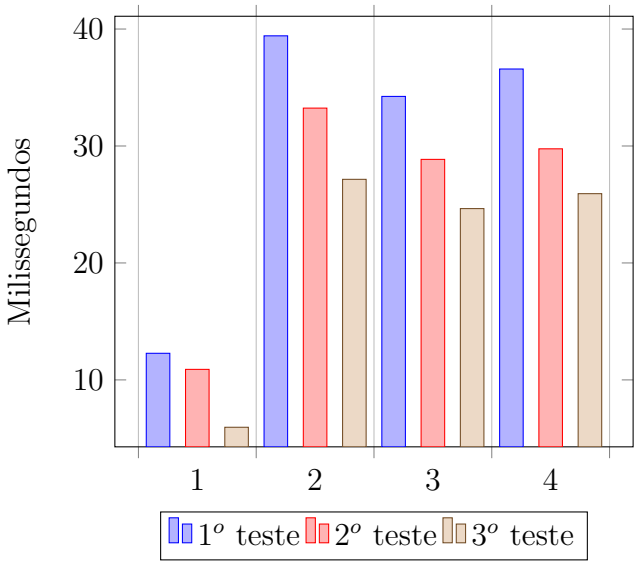


Figura 49: Histograma do Profiling dos 4 modelos implementados (1-Lambert 2-Oren-Nayar 3-Phong 4-Cook-Torrance).

Este gráfico exibe o que acontece com as médias de tempo de processamento para os quatro modelos, a medida que se varia a quantidade de unidades de processamento. As colunas em azul, são os tempos médios utilizando o máximo de recursos que dão suporte ao paralelismo da GPU, correspodentes ao tempo médio dos modelos do primeiro teste. Nas colunas em vermelho, o tempo médio dos testes utilizando metade do poder de processamento paralelo da GPU, correspondentes ao tempo médio dos modelos do segundo teste e, por fim, em marrom, equivalentes ao tempo médio dos modelos no terceiro teste. A medida que a quantidade de unidades crescem, o tempo médio cresce quase que numa proporção constante.

Com estes testes, pode ser verificado que, mesmo modificando a quantidade de unidades de processamento, a diferença de tempo entre os modelos difusos são mais significantes do que a diferença de tempo entre os modelos especulares, que tem sua medida de performance em milissegundos sempre muito aproximadas.

6 CONCLUSÕES E TRABALHOS FUTUROS

É relevante analisar a diferença de tempo entre os modelos, mesmo que esta diferença, por ser muito pequena (menos de meia centena de milissegundos), possa parecer desconsiderável. Mas, dentro do contexto do *Path Tracing* na GPU, essa diferença pode vir a reduzir consideravelmente a performance da aplicação, por exemplo, considerando um renderer de resolução 512x512, com 16 amostras por pixel, 5 chamadas recursivas (grau de profundidade constante igual a 5 de recursão) e tendo consciência que cada amostra de cada pixel é executada por um *work-item* do integrador e que cada chamada recursiva irá executar o cálculo da BRDF correspondente aos *kernels* do *profiler* desenvolvido nos testes. Além disto, a diferença de tempo entre os modelos, medida no mesmo *profiler*, aparentemente desconsiderável, pode se tornar ainda mais relevante se a GPU da placa gráfica utilizada para executar o algorítmo tenha um número de *work-groups* e *work-items* reduzidos. Resumindo: não há uma resposta fixa sobre qual modelo é melhor neste contexto, porque a escolha de qual modelo a implementar é relativa e depende de dois fatores: os recursos utilizados (qual a configuração da GPU utilizada) e também depende do que importa mais, a performance ou o realismo da imagem sintetizada pela aplicação na GPU.

Deste modo, diante dos resultados obtidos nesta pesquisa, é possível concluir que: nas situações em que o realismo é mais relevante, o melhor modelo difuso a se implementar é o modelo de Oren-Nayar, porque ele simula uma maior diversidade materiais difusos, uma vez que simula materiais difusos rugosos, além de apenas simular superfícies perfeitamente polidas (restrição do modelo de Lambert, em relação à Oren-Nayar) e o melhor modelo especular é o de Cook-Torrance, por haver um maior rigor a respeto das leis de conservação de energia que simulam o transporte de energia e Phong ser um modelo empírico. Mesmo em situações em que não haja tanta preocupação com as leis da física e do transporte de energia e seja interessante apenas a estética, ou a aparência visual dos resultados, mesmo sabendo que o modelo de Phong poderá ser considerado uma boa aproximação do modelo de Cook-Torrance um pouco menos custoso, o modelo de Cook-Torrance ainda permanece como a melhor opção, pela mesma razão da situação de conflito de escolha verificada acima. Já nos casos em que a performance for mais importante do que a qualidade visual dos resultados dos modelos, como se verificou na sessão de *profiling*, o melhor modelo difuso é o modelo de Lambert e o melhor modelo especular é o modelo de Cook-Torrance (apesar de ser um modelo especular baseado em microfaces, com a utilização da aproximação de Schilick [23], torna-se uma solução de custo bem aproximado ao modelo de Phong, que é um modelo empírico. Como Cook-Torrance é um modelo baseado em física, torna-se conveniente utilizá-lo, já que alcança resultados melhores, ao preço de uma diferença de tempo pequena). Com isso, atingimos o objetivo principal deste trabalho: oferecer ao leitor uma análise dos modelos e concluir,

em situações de conflito de escolha, qual o modelo mais interessante de ser implementado (dentre os que foram apresentados e estudados).

Além disso, este documento contempla, com uma certa Multidisciplinaridade, os pontos mais importantes que envolvem os materiais no contexto do *Path Tracing* na GPU, discutindo assuntos como radiometria e probabilidade, devidamente referenciados com bibliografias que abordam estes temas no contexto do algoritmo. Também contempla os principais modelos de função de distribuição de reflectância bidirecional utilizados atualmente e cita outros modelos que não foram discutidos, oferecendo ao leitor uma boa fonte de estudos e referências para quem tiver interesse em se aprofundar no algoritmo e nos modelos e eventualmente implementá-los, ou para aqueles que estão à procura do que há sobre este tema na literatura. Além disso, discute-se também a performance de cada modelo que foi apresentado em maior profundidade e comparando os modelos difusos e especulares entre si, dando ao leitor uma boa análise dos modelos, possibilitando a discussão da decisão de qual modelo, seja ele difuso ou especular, deve ser implementado, dentre os modelos citados (já que existem outros modelos difusos e especulares a serem estudados e implemnentados), a depender da situação.

Este trabalho também pode originar futuros estudos, como a avaliação de performance de alguns modelos que foram citados, que não foram avaliados, como o modelo difuso de difuso de Opik(modelo empírico difuso, que pode substituir a equação de Oren-Nayar para simular o brilho da lua em aplicações que não haja tanta relevancia quanto à plausabilidade da BRDF), ou o modelo anisotrópico de Cook-Torrance, como também a inclusão de vários outros modelos dentro da infinidade de modelos que não foram citados, como por exemplo alguns dos que foram elencados e classificados em [18]. Além disso, algumas aproximações e maneiras diferentes de calcular os termos D, F e G da equação de reflectância de Cook-Torrance foi apresentada neste estudo, o que também ocorre com o modelo de Oren-Nayar. Estas equações podem originar estudos aprofundados semelhantes a este, analisando performance e realismo de cada variação em maior nível de detalhe, a fim de investigar a melhor forma de expressar estas equações para renderizadores em tempo real. Além disso, muito brevemente, foi abordado no presente documento, uma explicação sobre amostragem por importância. Uma boa sugestão de trabalho futuro seria analisar a amostragem por importância, comparar os resultados com a amostragem uniforme em outros modelos, analisar performance da amostragem, fazendo um trabalho semelhante a este, só que com um foco na amostragem por importância, não apenas no cálculo das BRDFs.

REFERÊNCIAS

[1] H. Christensen, Per and Jarosz, Wojciech. **Foundations and Trends® in Computer Graphics and Vision**, vol. 10, No. 2, pp 103-175, Outubro, 2016.

[2] Tolkien, John Ronald Reuel **O Senhor dos Anéis: O retorno do rei**. Martins Editora, 2017.

[3] M. Sobol', Llya. **A Primer for the Monte Carlo Method**. CRC press, 1994.

[4] Marschner, Steve and Shirley, Peter. **Fundamentals of computer graphics, Fourth Edition**. CRC Press, 2015.

[5] Glassner, Andrew S. **Principles of Digital Image Synthesis**. Elsevier, 2014.

[6] Shirley, Peter and Morley, R Keith. **Realistic ray tracing**. AK Peters, Ltd., 2008.

[7] Schlick, Christophe. **An inexpensive BRDF model for physically-based rendering**. Wiley Online Library, 1994.

[8] Lagarde, S and Rousiers, CD. **Moving frostbite to physically based rendering**. SIGGRAPH 2014 Conference, Vancouver, 2014.

[9] Goral, Cindy M and Torrance, Kenneth E and Greenberg, Donald P and Battaile, Bennett. **Modeling the interaction of light between diffuse surfaces**. ACM SIGGRAPH computer graphics, ACM, vol. 18, No.3, pp 213-222, 1984.

[10] Grosser, Tobias and Zheng, Hongbin and Aloor, Raghesh and Simbürger, Andreas and Größlinger, Armin and Pouchet, Louis-Noël. **Polly-Polyhedral optimization in LLVM** in Proceedings of the First International Workshop on Polyhedral Compilation Techniques (IMPACT), vol. 2011, pp 1, 2011.

[11] Pharr, Matt and Jakob, Wenzel and Humphreys, Greg. **Physically based rendering: From theory to implementation**. Morgan Kaufmann, 2016.

[12] Walter, Bruce and Drettakis, George and Parker, Steven. **Interactive rendering using the render cache** in Rendering techniques' 99, pp 19-30, 1999.

[13] Bittner, Jiří and Hapala, Michal and Havran, Vlastimil. **Fast insertion-based optimization of bounding volume hierarchies** in Computer Graphics Forum, Wiley Online Library, vol. 32, No.1, pp 85-100, 2013.

[14] Shirley, Peter and Morley, R Keith and Sloan, Peter-Pike and Wyman, Chris. **Basics of Physically-based Rendering** ACM, November, 2012.

[15] Shirley, Peter. **Notes on Basic Radiometry and BRDF Models**.

[16] De Greve, Bram. **Reflections and refractions** in ray tracing, Retrived Oct, 2006.

[17] Gotanda, Yoshiharu. **Designing Reflectance Models for New Consoles** ACM, November, 2012.

[18] Montes Soldado, Rosana and Ureña Almagro, Carlos. **An Overview of BRDF Models**, 2012.

[19] Tay, Raymond. **OpenCL parallel programming development cookbook** Packt Publishing Ltd, 2013.

[20] Lambert, JH. **Photometria sive de mensure et gradibus luminis colorum et umbra**Eberhard Klett, 1760.

[21] Matusik, Wojciech and Pfister, Hanspeter and Brand, Matt and McMillan, Leonard. **A Data-Driven Reflectance Model in ACM Transactions on Graphics**, vol. 22, No. 3, pp 759-769, Julho, 2003.

[22] **MERL BRDF Database**, 2006. Disponível em: <https://www.merl.com/brdf/>. Acesso em: 24/04/2018.

[23] Hoffman, Naty. Background: physics and math of shading.**Physically Based Shading in Theory and Practice**, vol. 24, No. 3, pp 221-223, 2013.

[24] Torrance, Kenneth E and Sparrow, Ephraim M. **Theory for off-specular reflection from roughened surfaces** in Josa, Springer, vol. 57, No. 9, pp 1105-1114, 1967.

[25] Walter, Bruce and Marschner, Stephen R and Li, Hongsong and Torrance, Kenneth E. **Microfacet models for refraction through rough surfaces** in Proceedings of the 18th Eurographics conference on Rendering Techniques, Eurographics Association, pp 195-206, 2007.

[26] Oren, Michael and Nayar, Shree K. **Generalization of Lambert's reflectance model** in Proceedings of the 21st annual conference on Computer graphics and interactive techniques, ACM, pp 227-251, 1994.

[27] Oren, Michael and Nayar, Shree K. **Generalization of the Lambertian model and implications for machine vision** in International Journal of Computer Vision, Springer, vol. 14, No. 3, pp 227-251, 1996.

[28] Wolff, Lawrence B and Nayar, Shree K and Oren, Michael. **Improved diffuse reflection models for computer vision** in International Journal of Computer Vision, ACM, vol. 30, No. 1, pp 55-71, 1998.

[29] Phong, Bui Tuong. **Illumination for computer generated pictures** in Communications of the ACM, ACM, vol. 18, No. 6, pp 311-317, 1975.

[30] Ashikhmin, Michael and Shirley, Peter. **An anisotropic phong BRDF model** in Journal of graphics tools, Taylor & Francis, vol. 5, No. 2, pp 25–32, 2000.

[31] Cook, Robert L and Torrance, Kenneth E. **A reflectance model for computer graphics** in ACM Transactions on Graphics (TOG), ACM, vol. 1, No. 1, pp 7-24, 1982.

[32] Kelemen, Csaba and Szirmay-Kalos, Laszlo. **A microfacet based coupled specular-matte BRDF model with importance sampling** in Eurographics Short Presentations, vol. 2, No. 3, pp 4, 2001.

[33] Öpik, Ernst. **Photometric measures on the moon and the earth-shine** in Publications of the Tartu Astrofizica Observatory, vol. 26, 1924.

[34] Shirley, Peter and Wang, Changyaw. **Direct lighting calculation by monte carlo integration** in Photorealistic Rendering in Computer Graphics, Springer, pp 52-59, 1994.

[35] Heasly, Benjamin S and Cottaris, Nicolas P and Lichtman, Daniel P and Xiao, Bei and Brainard, David H. **RenderToolbox3: MATLAB tools that facilitate physically based stimulus rendering for vision research** in Journal of Vision, vol. 14, No. 2, pp 6, 2014.

[36] Scarpino, Matthew. **OpenCL in action** in Manning Shelter Island, 2012.

[37] Hanrahan, Pat and Krueger, Wolfgang. **Reflection from layered surfaces due to subsurface scattering** in Proceedings of the 20th annual conference on Computer graphics and interactive techniques, ACM, pp 165-174, 1993.

[38] Jensen, Henrik Wann and Marschner, Stephen R and Levoy, Marc and Hanrahan, Pat. **A practical model for subsurface light transport** in Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, pp 511-518, 2001.

[39] Kajiya, James T. **The rendering equation** in ACM SIGGRAPH computer graphics, ACM, vol.20, No.4, pp 143-150, 1986.

[40] Christen, Martin. **Ray tracing on GPU** in Master's thesis, Univ. of Applied Sciences Basel (FHBB), vol.19, Jan, 2005.

[41] Jacobs, Gabriel and Ip, Barry. **Establishing user requirements: incorporating gamer preferences into interactive games design** in Design Studies, Elsevier, vol.26, no.3, pp 243-255, 2005.

[42] Tsakok, John A. **Faster incoherent rays: Multi-BVH ray stream tracing** in Proceedings of the Conference on High Performance Graphics 2009, ACM, vol.26, no.3, pp 151-158, 2009.

[43] McCool, Michael D. **Anisotropic diffusion for Monte Carlo noise reduction** in ACM Transactions on Graphics (TOG), ACM, vol.18, No.2, pp 171-194, 1999.

[44] Joule, James Prescott. **XVII. On the effects of magnetism upon the dimensions of iron and steel bars in The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science**, Taylor & Francis, vol.30, pp 76-87, 1847.

[45] Jakob, Wenzel. **Página de download e documentação do Mitsuba Renderer**. Suiça: Mitsuba-Renderer.org, 2010. Disponível em: <http://www.mitsuba-renderer.org/>. Acesso em: dia 31/04/2019.

[46] Eaton, John W. **Página de download e documentação do Octave**: GNU, 1998-2019. Disponível em: <https://www.gnu.org/software/octave/>. Acesso em: dia 31/04/2019.

[47] **Página da documentação do OpenCL**: The Khronos Group Inc., 2007-2010. Disponível em: <https://www.khronos.org/registry/OpenCL/sdk/1.1/docs/man/xhtml/>. Acesso em: dia 31/04/2019.

[48] **Página de acesso aos códigos-fonte do livro OpenCL in Action**: Manning Publications Co, 2019. Disponível em: <https://www.manning.com/books/opencl-in-action>. Acesso em: dia 31/04/2019.

[49] **página de download e documentação do CUDA**: NVIDIA Corporation®, 2019. Disponível em: <https://developer.nvidia.com/cuda-downloads>. Acesso em: dia 31/04/2019.

[50] **Página de download e documentação do Intel® Graphics Performance Analyzers**: Intel®, 2019. Disponível em: <https://software.intel.com/en-us/gpa>. Acesso em: dia 31/04/2019.